(2)

AD____________________
TECOM Project No. 7-CO-M91-EPD-005
Publication No. USAEPG-FR-1425

METHODOLOGY INVESTIGATION

FINAL REPORT

AUTOMATED REAL-TIME TEST SCENARIO GENERATION

PHASE I

BY

CURTIS MASSIE

C³I Instrumentation Division
Test Support Directorate

US ARMY ELECTRONIC PROVING GROUND
FORT HUACHUCA, ARIZONA 85613-7110

OCTOBER 1991

PREPARED FOR: US Army Test and Evaluation Command
Aberdeen Proving Ground, MD 21005-5055

US Army Test and Evaluation Command
Aberdeen Proving Ground, MD 21005-5055

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date Jun 30 1986

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) USAEPG-FR-1425	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION USA Electronic Proving Ground 6b OFFICE SYMBOL (If applicable) STEEP-TS-C	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Fort Huachcua, AZ 85613-7110	7b ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION USA Test & Eval Command 8b OFFICE SYMBOL (If applicable) AMSTE-TC-D	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Aberdeen Proving Ground, MD 21005-5055	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO | PROJECT NO | TASK NO | WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Methodology Investigati Final Report, Automated Real-Time Scenario Generation, Phase I (UNCLASSIFIED)

12 PERSONAL AUTHOR(S)
Curtis Massie

13a TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
	FROM Oct 90 TO Oct 91	October 1991	88

16 SUPPLEMENTARY NOTATION

17 COSATI CODES

FIELD	GROUP	SUB-GROUP
09	02	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Automated scenario generation, message-oriented test, multiple nodes, test driver, time-ordered event lists (TOELs), traffic-orietned test, graphical user interface.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)
This report covered Phase I of an investigation to identify flexible and powerful designs for automated real-time test scenario test generation. This report will support development of new capabilities in the USAEPG family of test drivers. Current drivers need to be more interactive--their scenarios need to change in response to messages received.
This report identified other design enhancements including the following: graphical user interface, time-ordered event lists (TOELs) for message-oriented tests, scenario scripting in terms of message traffic loading for traffic-oriented tests, support for scripting an integrated scenario involving multiple nodes (instead of scripting each node separately), and an interface with an exercise driver (like the Corps Battle Simulator).

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code) | 22c OFFICE SYMBOL

DEPARTMENT OF THE ARMY
HEADQUARTERS, U.S. ARMY TEST AND EVALUATION COMMAND
ABERDEEN PROVING GROUND, MARYLAND 21005-5055

REPLY TO
ATTENTION OF

AMSTE-TC-D (70-10p) 17 Mar 92

MEMORANDUM FOR Commander, U.S. Army Electronic Proving Ground,
ATTN: STEEP-TS-C, Fort Huachuca, AZ 85613-7110

SUBJECT: Methodology Investigation Final Report, Automated Real-Time Test Scenario Generation, Phase I, TECOM Project No. 7-CO-M91-EPD-005

1. Subject report is approved.

2. Point of contact at this headquarters is Mr. James Piro, AMSTE-TC-D, amstetcd@apg-9.apg.army.mil, DSN 298-2170.

FOR THE COMMANDER:

[illegible]

FREDERICK D. MABANTA
Chief, Tech Dev Div
Directorate for Technology

A-1

Table of Contents

Please use this space for your notes

Foreword

This report represents the completion of Phase I of the Automated Real-time Scenario Generation investigation and defines the basic requirements for a new Automated Real-time Scenario Generation System.

This report was developed in accordance with U.S. Army Test and Evaluation Command (TECOM) Regulation 70-17. Its original objective as defined in the Methodology Investigation Proposal has been broadened to include defining requirements for an entire scenario generation design.

In Phase II of the investigation the basic requirements and high-level system design defined here will be used to create a prototype of the Automated Real-time Test Scenario Generation System. Further study of requirements will be completed. Answers to unresolved issues will be included in a second methodology report to be completed by the end of Phase II.

The author gratefully acknowledges those who contributed to this report including the following:

Wells Aamodt
Gwen Gillen

Also, the author gratefully acknowledges those who agreed to be interviewed and those who extended cooperation, especially the following:

Steve Bortman
Lee Corrington
Bob Delashmitt
Scott Edwards
Tom Flahie
Paul Fowler
Dale Fulton
Don Hand
Jim Hawk

Nancy Helton
Hank Hockenberger
Pat Kerr
Martin McCord
MAJ Dave McClung
Francine Moore
Randy Schultz
Ken Van Karsen
Dennis Wilker

Please use this space for your notes

1. Introduction

1.1 Background

The background as stated in the Methodology Investigation Proposal (MIP) (provided at Appendix A) is quoted as follows: The Micro-Test Item Stimulator (Micro-TIS) is a test driver which provides messages to a system under test (SUT). The test driver currently responds to incoming messages from the SUT with prescribed responses. A method of automated interaction with the SUT is required.

NOTE: A list of the acronyms and abbreviations used throughout this report is provided at Appendix B. A glossary of key terms used throughout this report is provided at Appendix C. A review of the terms used should help considerably with understanding the contents of this report.

1.2 Problem

The problem as stated in the MIP is illustrated in Figure 1 and is quoted as follows: Changing the Micro-TIS test driver's prescribed responses is an untimely process. The test driver does not have the capability to respond to a message in the same manner as the SUT responds. The test driver cannot participate in the interactive exercises except as noted above.

The current pretest software for the Micro-TIS was designed to create a simple time-ordered list of messages to be sent to the SUT. A major requirement during development was test repeatability, which resulted in a lack of interactive capabilities.

These limitations have made our current Pretest system inadequate for some testing situations. New systems will require support for traffic-oriented tests, interactive scenario execution, and the capability to respond to the SUT during realtime.

Scripting is a labor intensive process that requires a high degree of expertise. The current methodologies result in high costs for test conduct.

1.3 Objective

The objective according to the MIP is quoted as follows: Determine a method for automated interaction of the Micro-TIS test driver with the SUT. Phase I will define the required automated scenario generation system. Phase II will create a prototype of that system and develop interfaces so it can be used real-time to respond to a message sent from the SUT.

As the investigation progressed, the objective was expanded as shown in Figure 2 and described in the following subparagraphs.

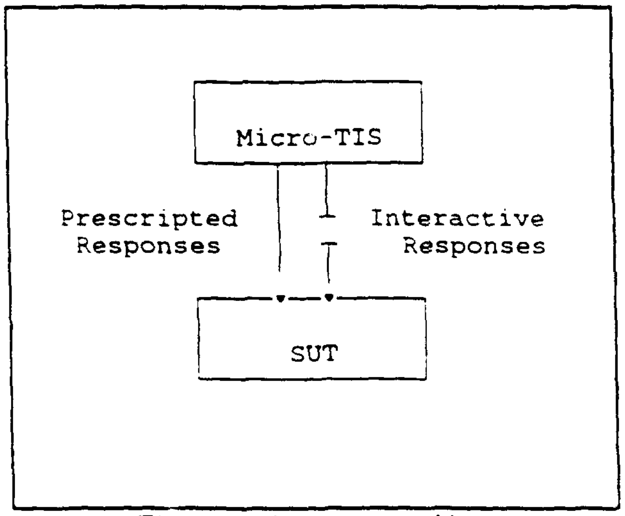


Figure 1. Response problem.

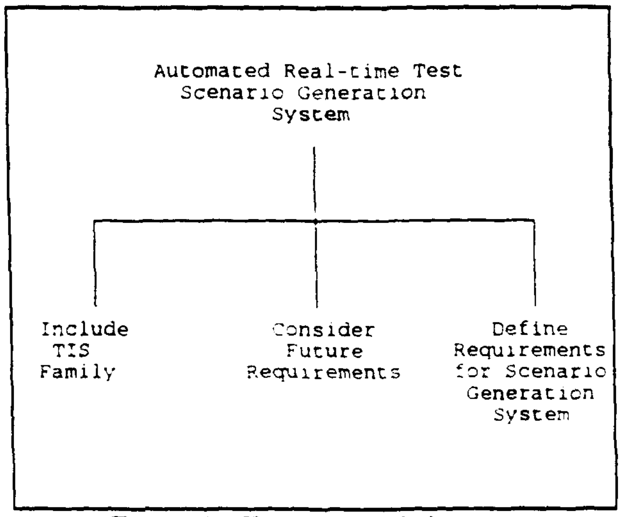

Automated Real-time Test Scenario Generation System
Include TIS Family
Consider Future Requirements
Define Requirements for Scenario Generation System

Figure 2. Expansion of objective.

• The Micro-TIS is one of a number of hardware/software platforms that make up the Test Item Stimulator (TIS) family. The Versa Module Eurocard-Test Item Stimulator (VME-TIS) is currently being developed and will gradually replace the Micro-TIS. Our investigation was expanded to address requirements for current and developmental platforms in the TIS family. Emphasis was placed on the VME-TIS system.

• To conduct as complete a study as possible, we studied requirements for future TIS systems as well.

• Originally the study was directed towards interactive real-time scenario generation. Since new hardware/software platforms are currently being developed that will require rewriting Pretest software, the scope of the investigation was broadened to include all automated scenario generation requirements. Emphasis will be on real-time interaction with the SUT.

1.4 Procedures

Our approach to the study was to initially study existing and developmental systems, then look at some systems in the planning stage, evaluate future requirements in general, and then to develop a list of general design requirements, appropriate methodologies, and feature enhancements that are needed in the new Automated Real-time Test Scenario Generation System. This approach is illustrated in Figure 3.

NOTE: A list of the documents reviewed during the study is provided in Appendix M.

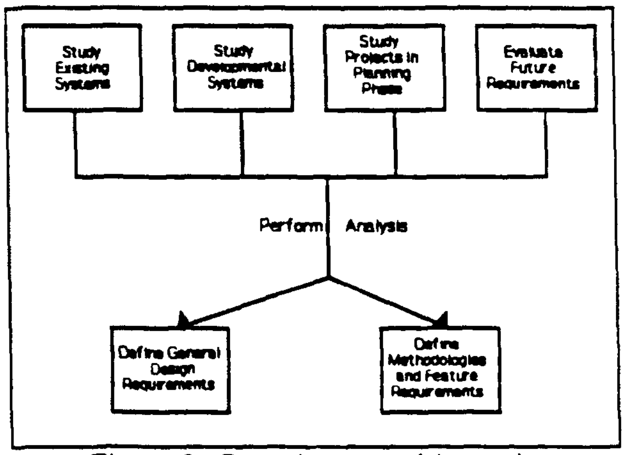

Study Existing Systems
Study Developmental Systems
Study Projects in Planning Phase
Evaluate Future Requirements
Perform Analysis
Define General Design Requirements
Define Methodologies and Feature Requirements

Figure 3. Procedures used in study.

1.4.1 Existing Systems

We first reviewed existing methods for scenario generation (including efforts involving TIS and Micro-TIS systems). We studied ongoing efforts for improving those methods. We documented ideas for improving current processes. The specific systems studied were:

• Enhanced Position Location Reporting System (EPLRS).

• Joint Tactical Information Distribution System (JTIDS).

• Mobile Subscriber Equipment (MSE).

• Portable TADIL Tester (PTT).

1.4.2 Developmental Systems

Next we looked at developmental systems to determine what changes to scenario generation were planned and what additional requirements had been identified.

• We focused on VME-TIS since that system is the "state-of-the-art" for the TIS family. The scheduled completion of the real-time and posttest developments for VME-TIS was September 1991. Since the pretest (scenario generation) development was scheduled to start in Fiscal Year (FY) 92, some of the results of this investigation could be incorporated into that development effort. A high level view of the VME-TIS System is shown in Figure 4. The implementations under development for the VME-TIS were as follows:

EPLRS.
JTIDS Data Reduction Applique (JDRA).

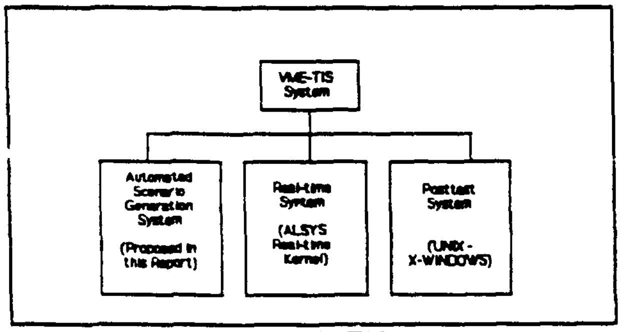


Figure 4. VME-TIS system.

• We also studied the Test Control Center (TCC) system which is currently under development and is geared towards providing control and management of multiple, remote instrumentation. It is intended to give a test manager control of all instrumented nodes for network-wide tests.

1.4.3 Projects in the Planning Stage.

We reviewed scenario generation requirements for five projects that are currently in the planning stage. We identified new requirements for those five projects and evaluated how those requirements would fit into the existing scenario generation software. Those projects were:

- Battlefield Functional Areas (BFA).
- Mobile Automated Instrumentation Suite (MAIS).
- MSE Option Year 4 (OY4).
- Tactical Deployment Simulation System (TADSS).
- Tactical Digital Information Link (TADIL) Analysis Tool (TAT).

1.4.4 Identifying Requirements, Methodologies, and Feature Enhancements

After considering the existing systems, developmental systems, systems in the planning stage, and future requirements in general, we pulled all the information together to develop a general idea of what the system design should be and determine what major enhancements to current scenario generation efforts should be implemented. Our emphasis was on finding more flexible/powerful designs for automated scenario generation that would allow development of new capabilities, including the generation and transmission in real-time of the correct response to a received message or instruction.

NOTE: Details related to the investigation are provided in Section 2.

1.5 Results

After studying the existing scenario generation systems, those under development, those in the planning stages, and considering future test efforts, we arrived at the following results.

1.5.1 System Types

The general types of systems that must be supported are described in the following subparagraphs.

Message-oriented Tests. Examples of this type of test are the TADIL A/B/J protocol tests. The main object is to test the protocols rather than the equipment, so message rates and throughput are relatively unimportant. The exact content of the messages is critical. Timing of messages is also critical. In our current implementations of the TADILs, a single scripted message may cause a number of messages to be sent in real-time due to protocol simulation in the system-specific applique (SSA).

Traffic-oriented Tests. Examples of this type of test are the EPLRS and MSE tests. Message rates and throughput are important. Content of message is not as important. Scripts are frequently described in terms of traffic rates and conditions, e.g., 900 messages per hour with an offhook rate of 10 percent. Scripts involve multiple nodes running simultaneous scenarios. Our current methodology proved inadequate for this type of test. An average three hour test might involve creating a script of 2700 nearly identical messages for each of dozens of nodes. EPLRS and MSE each developed a stand alone, off-line system to assist in generating their scenarios.

1.5.2 Single Design Approach

A single basic approach to the design must be used since resources can only support one approach towards improving automated scenario generation. A single design approach must be established to efficiently support both message-oriented and traffic-oriented tests.

1.5.3 General Design Requirements

The following are general requirements that should be incorporated in the high-level system design:

- Maximized Reuse of Developed Software.
- Off-the-Shelf Software.
- Real-time Integration.
- Development Using Computer Aided Software Engineering (CASE) Tool.

NOTE: More detailed information on these requirements is provided in paragraph 1.6 below.

1.5.4 Applicable Methodologies

In order to support the design requirement of software reuse, an object oriented development methodology should be the methodology used for this development.

1.5.5 Data Collected

Much of the data collected during this study has been included in the appendices to this report. A list of the appendices included is provided at Figure 5.

- Appendix A - Methodology Investigation Proposal
- Appendix B - Acronyms and Abbreviations
- Appendix C - Glossary
- Appendix D - CBS Interfaces
- Appendix E - EPLRS Data Base Structures
- Appendix F - Existing Scenario Generation Software for MSE
- Appendix G - General Description of VME-TIS System
- Appendix H - VME-TIS Communications Module
- Appendix I - Extract from IDD for ADDS Implementation of VME-TIS
- Appendix J - Extract from Instrumentation Validation Test Plan for TCC
- Appendix K - Study of MSE System for OY4 Testing
- Appendix L - Information on TADSS
- Appendix M - References
- Appendix N - Distribution List

Figure 5. List of appendices.

1.6 Analysis

Our analysis of the assembled information was geared toward determining what general design requirements, methodologies, and feature enhancements should be combined to produce the type of automated scenario generation system needed.

1.6.1 System Components

As illustrated in Figure 6, the Automated Real-time Test Scenario Generation System will consist of two parts -- a Scenario Creation capability and a Scenario Execution capability. The Scenario Creation capability will be associated with pretest operations. The Scenario Execution capability will be associated with real-time operations.

1.6.2 General Design Requirements

Several general requirements should be incorporated rated in the high-level system design. An analysis of the reasons for the inclusion of each requirement in the design is provided in the following subparagraphs.

Maximized of Developed Software. Programming should avoid system-specific code. Table or

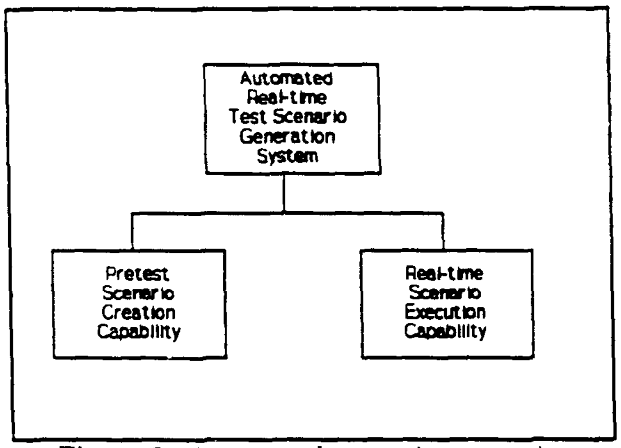

Automated Real-time Test Scenario Generation System
Pretest Scenario Creation Capability
Real-time Scenario Execution Capability

Figure 6. Automated scenario generation system components.

data base driven software allows the program to be adapted to a new system without changing code.

Off-the-Shelf Software. To save time and reduce costs, we should use off-the-shelf software. There are packages available for graphics management, prototyping, communications, data base management systems, etc. The available products should be studied during the design phase.

Integration. Considerable integration with other testing functions is required. A decision should be made early in the design process on the degree of integration of scenario generation, posttest and real time.

- VME-TIS Real-time and Posttest Systems exist and the cost of integration needs to be estimated. Many enhancements are by their nature real-time functions. A multiple CPU solution may require enhancing real-time operations to support Ethernet.
- An illustration of one possible VME-TIS CPU configuration is provided at Figure 7. Another possibility would be having the Scenario Generation CPU run on a separate Pretest platform. A third possibility would be to have the Scenario Generation and Posttest software run on the same CPU on the real-time platform, which would result in all the CPUs being run on the same platform.

Development Using CASE Tools. CASE tools should be used to aid in the development of the Automated Real-time Test Scenario Generation System. CASE tools provide a central location

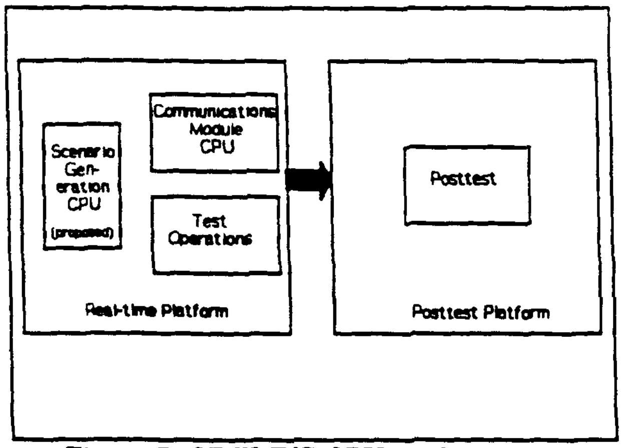


Figure 7. VME-TIS CPU configuration.

for design information and make coordination and communication on the project simpler. Many CASE tools automate the task of providing documentation to DOD-STD-2167A.

1.6.3 Applicable Methodologies

An object oriented development methodology should be used in the Automated Real-time Test Scenario Generation System. Object oriented techniques should be used where possible since those methods provide a clean path for customization and tailoring of systems by developing objects of varying complexity as required.

1.6.4 Feature Enhancements

A number of feature enhancements are needed for incorporation into the high-level system design. An analysis of the reasons for the inclusion of each item in the design and an identification of the corresponding system capability(ies) are provided in the following subparagraphs.

Improved User Interface. The standard for user interfaces has advanced since the TIS family was designed. The current interface is character- and menu-oriented and requires a high level of user expertise. Useability would be improved if we moved to a more graphical, pointer driven interface. Many scripting tasks would be better served by a graphical interface, e.g., track placement, network definition.

Support for Traffic-oriented Tests. We have already encountered tests which required development of separate scenario scripting systems because the generic TIS does not support traffic-oriented tests. We need to support scripting in terms of loading. It should be possible to define scenarios in terms of message rates/conditions in addition to the current simple lists.

Time Ordered Event Lists

• A time-ordered event list (TOEL) is a list of simple messages which are sent under control of real-time execution. The differences between our existing Pretest System and a new system using TOELs are illustrated in Figure 8 and Figure 9 and described in the following subparagraphs.

• Currently, a scenario is made up of a series of timed events. Events are made up of a series of timed messages. A scenario file is created placing each message for each event into the file according to its calculated scenario time. This process of expanding events is called elaboration. The scenario file is used as input to the real-time scenario execution process. Events are eliminated during the event elaboration process so real-time scenario execution sees a simple list of timed messages.

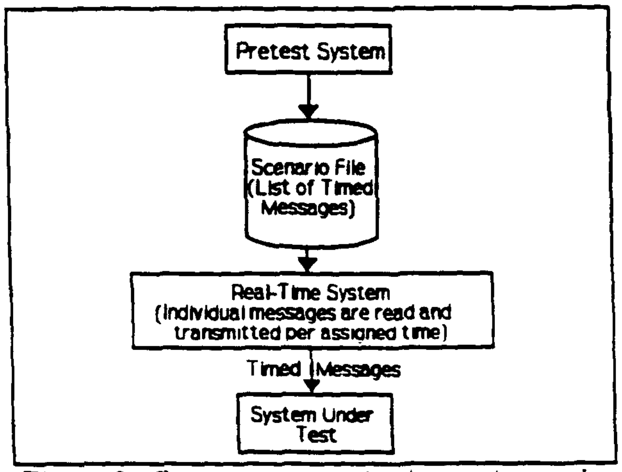

Pretest System
Scenario File (List of Timed Messages)
Real-Time System (Individual messages are read and transmitted per assigned time)
Timed Messages
System Under Test

Figure 8. Scenario generation / execution under current system.

• In a TOEL, the scenario is a list of messages and TOELs and the TOELs are elaborated by the real-time system. A TOEL event could be a simple list of messages (like our current events) or a more complex system with parameters and conditional statements. Advance Field Artillery Data System (AFATDS), Maneuver Control System (MCS), and Army Tactical Command and Control System (ATCCS), would use this capability.

Interactive Scenario Execution. Operational testing is interactive and user driven. Users determine events based on the current situation.

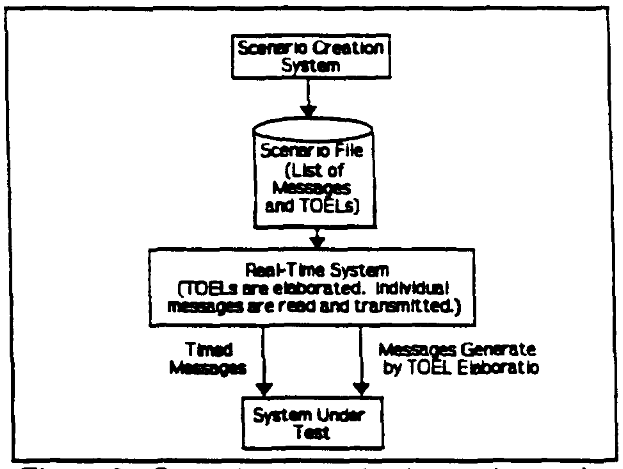

Scenario Creation System
Scenario File (List of Messages and TOELs)
Real-Time System (TOELs are elaborated. Individual messages are read and transmitted.)
Timed Messages
Messages Generate by TOEL Elaboratio
System Under Test

Figure 9. Scenario generation / execution under new system.

Users want to compose, execute, and modify scenarios during real time. We need better control over scenario execution in real time. The TAT system, for example, anticipates a very interactive environment with scenario execution controls such as single stepping, jumps, and manual control.

Simulation Support. Some systems require a simulation capability. Our Joint Tactical Command, Control, and Communications Agency (JTC^{3}A) TADIL-A/B/J systems currently have that capability to a limited extent. Simulation can be done to a number of levels.

Response Table. Often the instrumentation must respond to the SUT. The responses can be provided by simulating parts of the protocols in the real-time system. A response table and trigger system also allow a system to respond appropriately. Requirements to respond to SUT activity during real-time execution are becoming more common.

Multi-node Scenarios

- Some systems involve testing in a network environment with many nodes. Currently each instrumented node must have its own scenario created independently.
- We need to be able to script an integrated scenario involving multiple nodes. This would simplify the scripter's task by allowing easier support of interactions between nodes. It could make more intelligent simulation possible by making activity on the network's other nodes visible to the simulator.

Central/Remote Control Support

- As stated in the preceding paragraph, many tests involve a multi-node network. Currently each instrumented node must have its own user. Some systems, e.g., ATCCS need to have a control center which can operate all the instrumented nodes. This would allow better coordination of the test. It would also reduce costs by eliminating the users at each node.
- The TCC system currently under development is intended to provide this type of support. The communications interface between the TCC and scenario generation should be defined early in the design phase.

Exercise Driver Interfaces

- There are a number of exercise drivers (war gaming systems) that we expect to see in use for testing and training. They may include Corps Battle Simulator/Joint Expert Support System (CBS/JESS), Deep Battle Integration Test System (DBITS), and Tactical Simulation (TACSIM).
- Interfacing with these systems would allow the instrumentation to be aware of and to respond to the exercise environment. This interface would improve the system's awareness of the environment.

1.7 Conclusions

USAEPG needs to create a more flexible and powerful automated scenario generation system in order to support present and future needs, especially related to generating and transmitting in real time the correct response to messages or instructions received from the SUT.

That system should be developed for the applicable members of the TIS family based on the general design requirements and methodologies detailed in paragraphs 1.5.3 and 1.5.4. A system incorporating those items will automatically generate scenarios, readily interface in real time with the SUT, accommodate development of future systems, and make better use of previous development efforts.

1.8 Recommendations

As a result of the Methodology Investigation Study, the recommendations described in the following subparagraphs are made:

Authorize Instrumentation Funding. It is recommended that the results of this study be approved and that we proceed to Phase II with instrumentation funds allocated to support the development effort plus further study.

Develop System Prototype. During Phase II a prototype of the Automated Real-time Test Scenario Generation System should be created to include the following: (1) the general design requirements defined in paragraph 1.5.3, and (2) the methodology identified in paragraph 1.5.4.

Complete Study of Outstanding Issues. Table 1 shows the issues that should be studied in depth. The results of that study should be incorporated into the Methodology Investigation Final Report for Phase II.

Table 1. Issues Needing Further Study

Capability	Comments
Message Traffic Generation Capability	Further study is needed on the use of a data base or tables during the generation effort.
Advanced TOELs Capability	Further design work will be required to determine whether advanced TOEL capabilities are desired and cost effective.
Automatic Interactive Execution Capability	Further study should be delayed until systems are identified that require the capability.
On-line Scenario Creation Capability	Further study will need to be conducted to resolve the incompatility with current real-time graphics capabilities.
General	Study available off-the-shelf software
Remote/Central Control Capability	Define the interface to the TCC
Simulation Capability	An investigation of the design of an elementary generic simulation capability such as response tables should be conducted.
Exercise Driver Interface Capability	Further study is needed to determine exactly what sort of communications the different exercise interfaces support. We will also need to define what the user will want to accomplish with these interfaces.
Multiple CPU Approach	Study adding Ethernet or another communication interface to the Real-time System

Please use this space for your notes

2.1 Initial Issues

We began the investigation by performing a general functional analysis, studying existing documentation to determine the current state of the Pretest System, looking at whether a one or two box approach would be most effective, and considering the advantages of environmentally based scenario generation.

2.1.1 Functional Analysis

We established the following basic functions as required in real time for the correct response to a received message or instruction to be generated and transmitted:

- Receive message.
- Decide that a response is required.
- Determine the correct response.
- Fill in the message fields correctly or branch to a pre-written response.
- Send response.

We determined that the following basic functions would be required prior to the real-time effort:

- Determine the protocol and extract a copy of every message for examination.
- Determine whether or not to respond and when to respond.
- Develop a table of responses.
- Develop tables of data or unfilled messages or of entire messages.

2.1.2 Current State of Pretest

We reviewed documentation of existing systems in the TIS family to determine the current state of the Pretest System. Currently pretest software generates individual messages. Although messages are generated as a group, the process is too time consuming to be executed during real-time test execution. Nevertheless, we thought it realistically possible to make incremental changes to an existing scenario during test execution.

2.1.3 Hardware Platform

We initially looked at the platform for the scenario generation software. The two possibilities identified were (1) creating a scenario creation system that runs on a separate box and interfaces with a TIS and (2) creating a full scenario generation system that would run with real time on the VME-TIS.

Two Box Concept. The two box concept seemed to avoid the complexities caused by the ongoing effort to change from Virtual Address Extension (VAX)-based Micro-TISs (that use FORTRAN) to Micro-based VME-TISs (that use Ada). Although this concept seemed less likely to be selected, we did investigate the following possibilities associated with it:

- In addition to using Ada, we considered developing a system using dBASE/CLIPPER.
- We looked at modifying an existing system designed for MSE. We tried using that system, Test Design Analysis Tool (TDAT), but found it complex and poorly documented, and our non-expert could not use it productively.

One Box Concept. We investigated integrating real-time scenario generation into the design of the VME-TIS capabilities. This was done in hopes of having a positive impact on that design. This was possible because lack of funding had delayed the development of the scenario generation capability for VME-TIS. The initially developed VME-TIS is using scenarios created on a Micro-TIS.

Two CPU Concept. We concluded that a better approach would be to go with a two CPU concept, with both CPUs within the same box. One CPU would be used to provide the Pretest Scenario Creation capability and the second CPU would be used to provide the Real-time Scenario Execution capability. An illustration of a VME-TIS System with Scenario Generation, Real-time, and Posttest CPUs included in the same box is provided at Figure 10.

2.1.4 Environmentally Based Scenario Generation

Under the environmentally based scenario generation approach, the environment is described graphically and the scenario generation software generates the scenario automatically.

Potential. We determined that the use of environmentally generated scenarios could reduce the time and expertise needed to generate an effective scenario and that the scenario could be changed during execution by changing the environment description. We also learned that environmentally based scenario generation could support an interface with CBS.

Corps Battle Simulator Interfaces. Information on the CBS interfaces has been excerpted from a MITRE paper, WP-90W00429, Automatic U.S. Message Text Format (USMTF) Text Output Messages (AUTOMsgs) Requirements Specification Document, dated October 1990 and is provided at Appendix D.

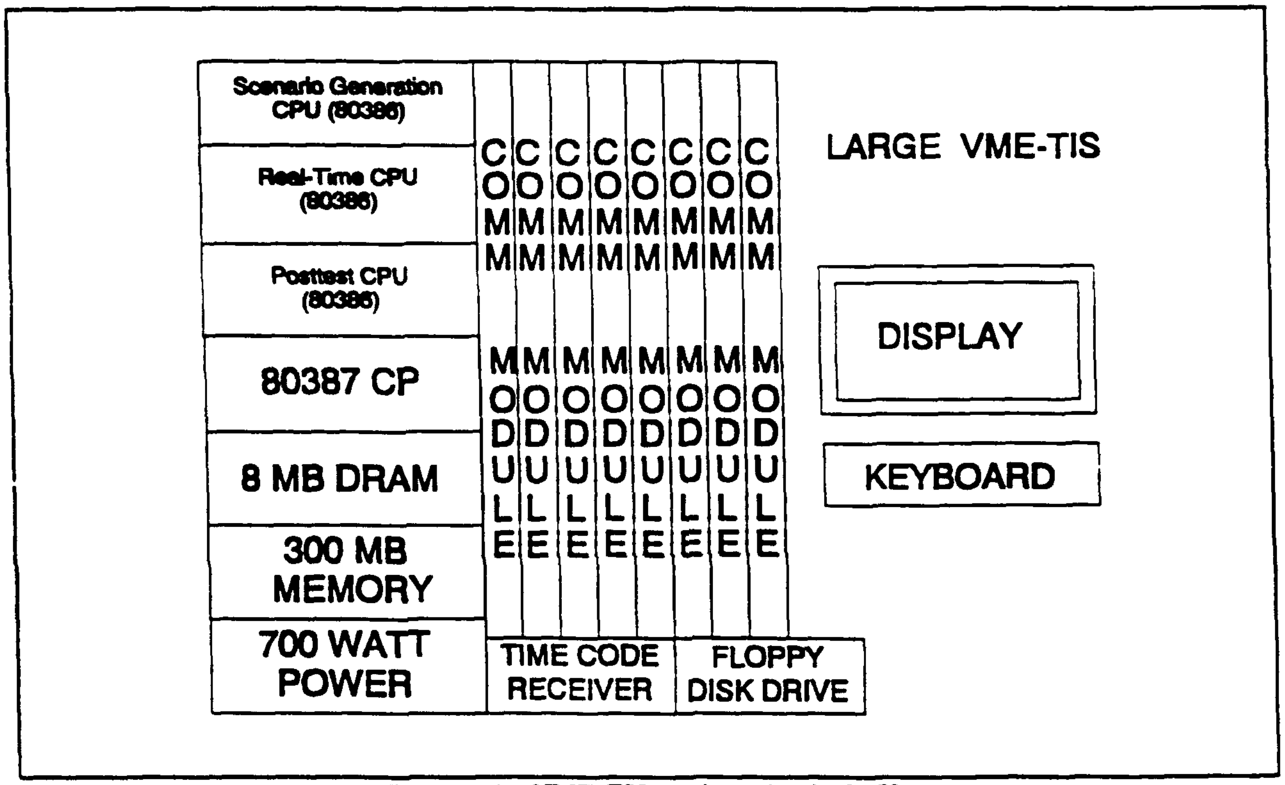


Figure 10. VME-TIS with multiple CPUs.

2.2 Existing Systems

We continued the investigation by studying existing systems that run at least partially on a MicroVAX platform using Micro-TIS software. The systems reviewed are described in the following subparagraphs.

2.2.1 Enhanced Position Location Reporting System

General System Description. EPLRS is a time division multiple access (TDMA) communications system with network control stations. Pretest scenario generation is run on a PC under DOS.

General Description of Scenario Generation. Developing traffic for an EPLRS test is a complex process. The current standard Pretest System proved inadequate; as a result, development of a separate scenario creation process for EPLRS has been necessary. That process is called XGEN. XGEN was written in C and is currently being redesigned to minimize manual errors and eliminate redundancy for the next test.

Operational Details

Step in Developing a Scenario. The following steps are taken as a scenario is developed:

- First, needlines are developed based on the ADDS requirements.
- The next step involves allocating timeslots which are big enough to handle the expected amount of traffic. This information is used by net control stations to control the network. This process uses some of the information contained in the data base records. Appendix E lists the data base tables used in this process.
- The XGEN program is run, which produces a schedule of individual messages which sum to the needline traffic rate. The messages are separated by a constant gap plus a random time. The randomness is a test requirement for EPLRS.
- In the redesigned process, a new requirement is to schedule several types of messages with different lengths and values in the speed-of-service data field which sum to the needline traffic rate.
- A utility program will print a list of the scheduled messages from XGEN for accuracy-checking. Since accuracy-checking the messages is an important part of the process, it could be redesigned as well.

Flow Diagram. A flow diagram of the process was developed and is provided at Figure 11. The first step shown is under development and will replace three existing steps -- WordPerfect text creation and editing, conversion to American Standard Code for Information Interchange (ASCII), and conversion to dBASE.

NOTE: In the redesigned process, data base records containing the needed information will be produced. These data base records will be contained in three data bases -- NEEDDB.DBF, UNITDB.DBF, and SITEDB.DBF. The structure of those data bases is shown in Appendix E.

Requirements Identified

Improved User Interface. The Pretest System created for EPLRS has no consistent user interface. A graphical interface and graphical scripting would simplify scripting and make training easier.

Support for Traffic-oriented Tests. The standard TIS Pretest System was not used because of lack of support for traffic-oriented tests. Support for message load scripting is a requirement for the new system.

Multi-node Scenarios. The Pretest System developed for EPLRS generated scenarios for many nodes from a single set of scenario definition files.

Central Control. A central control capability would have been useful. Test control was handled through voice communication with the users.

2.2.2 Joint Tactical Information Distribution System

General System Description. The JTIDS

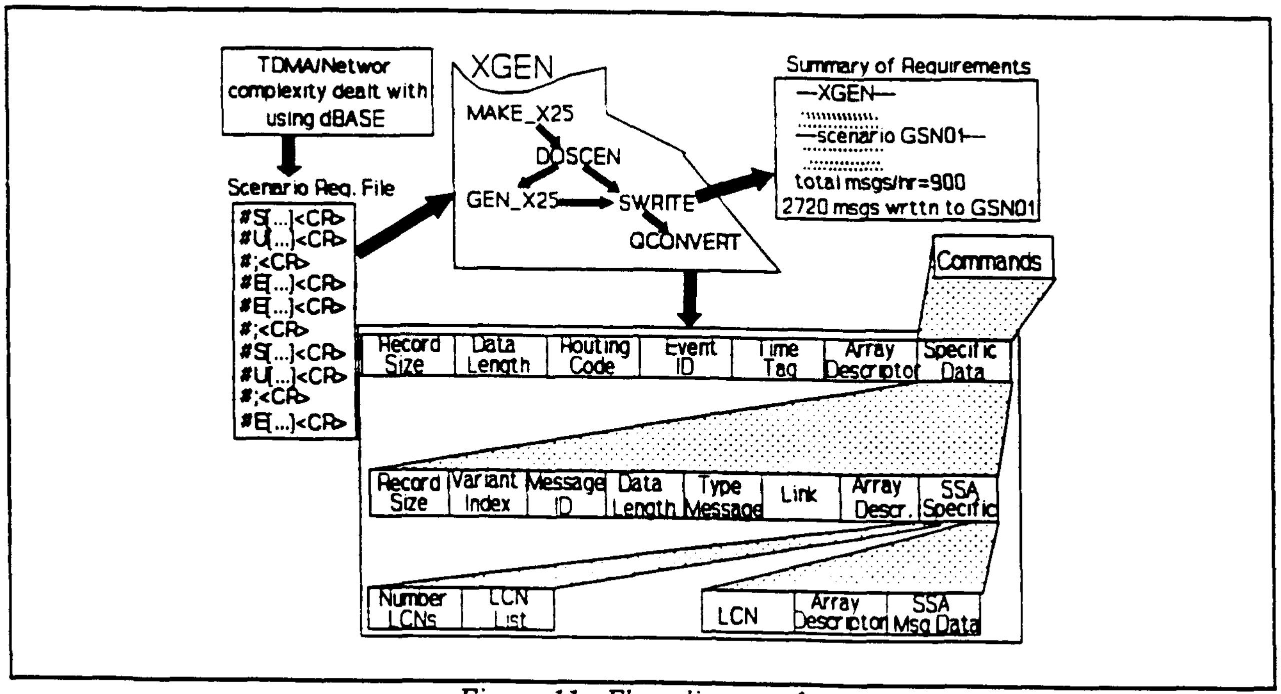


Figure 11. Flow diagram for scenario development.

instrumentation is a TADIL-J test system. It is designed to test equipment, not communications protocols. It is run on a combination of Micro-TISs and TISs.

General Description of Scenario Generation. The standard Micro-TIS Pretest System is used for scenario creation. Some scenarios are traffic oriented, repeating the same sequence of messages. Other scenarios are message oriented with emphasis on a specific sequence of messages and responses.

Operational Details

• The standard Pretest System is used. The Pretest System was modified to generate a unique, sequential, message identifier which is included in the data portion of each message. Both message- and traffic-oriented events are required. A traffic-oriented scenario might consist of an event with one second's worth of message traffic, repeated 7200 times to give a two hour scenario.

• Events are kept as small as possible to keep generation time down. One attempt to generate a 4-minute event ran for more than 24 hours before the process was canceled. Messages are at a high level of granularity. A message consists of one or more sets of TADIL-J codewords, network commands, etc.

Requirements Identified

Support for Traffic-oriented Tests. Some JTIDS scenarios are traffic oriented. Support for message loading would be useful.

TOELs. Scenarios sometimes repeat the same event over and over. Such events would benefit from the use of TOELs.

2.2.3 Mobile Subscriber Equipment System

General System Description. MSE is a voice and data network communications system. The Pretest System for MSE is run on a combination of Micro-TISs and TISs.

General Description of Scenario Generation. Currently there are two systems used to generate scenarios for testing MSE systems. The first was written in FORTRAN, runs on the TIS, and uses the basic pretest software designed to handle multiple SSAs. The second was written in PASCAL, runs on a Micro-TIS, and has been specifically designed to handle

MSE requirements only.

Operational Details. The details on operation of the pretest software and the TDAT software are provided in Appendix F.

NOTE: In addition to the two categories of software described here, another MSE scenario generation system was under development at the time of this investigation. It is titled MSE OY4 and is described in paragraph 2.4.3.

Requirements Identified

Improved User Interface. An improved user interface would be useful to MSE. Graphical scripting would simplify network definition and scenario creation.

Support for Traffic-oriented Tests. The TDAT Scenario Scripting System was created because of the lack of traffic-oriented support. MSE requires this feature.

Multi-node Scenarios. The MSE tests use multiple scenarios run simultaneously on various nodes on the network.

Central Control. A central/remote control capability would be useful.

2.2.4 Portable TADIL Tester System

General System Description. The PTT system was designed to test for the JTC^{3}A to test TADIL-A, TADIL-B, and TADIL-J protocols rather than equipment. It is run on Micro-TISs.

General Description of Scenario Generation. The standard Micro-TIS Pretest System is used for scenario generation. Tests are message-oriented with emphasis on testing protocols.

Operational Details. The standard Pretest System is used. Scenarios tend to be relatively short. Scenarios are usually made up of events which test a single protocol feature, separated by hold commands. The granularity of testing is relatively fine. Messages consist of a single codeword. Field content and validation is a major consideration. There are requirements to respond appropriately to the SUT. Much of the TADIL protocols are emulated by the real-time SSA to provide this capability. The scenario only scripts the original message. Redundancies are generated during real time through the protocol emulation.

Requirements Identified

Improved User Interface. A graphical interface and graphical scripting would simplify user training and make scripting more intuitive.

TOELs. Advanced TOELs could be useful in modularizing scenarios. It would allow a TOEL to be reused in various circumstances.

Interactive Scenario Execution. Scenarios are usually executed in small pieces with results examined during real time. There is a desire for free play testing capabilities.

Simulation. The system as currently implemented simulates much of the TADIL protocols. The users would like to extend this capability.

Responding to the SUT. The system requires timely, appropriate responses to messages seen from the SUT. This capability is currently provided by the Protocol Simulation capability.

Please use this space for your notes

2.3 Developmental Systems

The developmental systems of importance to this study were identified as the VME-TIS System and the TCC System.

2.3.1 Versa Module Eurocard-TIS System

General System Description. VME-TIS is the latest member of the TIS family of digital data link drivers. TISs are used to sti ıulate and test the digital data links of C^{3}I systems. TISs substitute for the systems with which the C^{3}I systems communicate and provide a total data-link test environment. Additional general information on the system is provided at Appendix G. Information on the communications module developed for VME-TIS is provided at Appendix H.

Individual Implementations. At the time of the development of this report, three implementations on the VME-TIS system were under development. They are as follows:

Enhanced Position Location Reporting System.

• A real-time SSA is under development for EPLRS for the VME-TIS. It uses the ALSYS real-time kernel as the operating environment. A Posttest Analysis System for VME-TIS EPLRS is under development. It uses SCO UNIX and MOTIF/X-WINDOWS as the operating environment. Both the Real-time and Posttest Systems run on Intel 386/486 IBM AT type systems. Scenario creation functions are being performed using the old EPLRS scenario generation routines. Some minor enhancements have been added.

• Information on scenario generation for EPLRS has been obtained from IDD-7A-10-00-00-OR DRAFT, Interface Design Document for the ADDS Implementation of the Versa Module Eurocard-Test Item Stimulator System, dated 10 May 91, and is provided at Appendix I.

JTIDS Data Reduction Applique. The JTIDS Data Reduction Applique (JDRA) is a posttest analysis system under development for the VME-TIS environment. It uses the SCO UNIX and MOTIF/X-WINDOWS environment. It is part of the same family of systems as the EPLRS VME-TIS Posttest System. Pretest and real-time support are provided by the Micro-TIS. The JTIDS Micro-TIS Pretest System is used for scenario development.

2.3.2 Test Control Center System

General System Description. The function of the TCC is to provide central/remote control for MSE testing. It is already providing a limited set of control functions for the Real-time Micro-TIS System. Its capabilities could be extended to provide the needed central/remote control capabilities. The communications interface between the TCC and the new Automated Real-time Test Scenario Generation System needs to be fully defined during the design phase of the new system.

General Description of Scenario Development. Not applicable.

Operational Details. The details of TCC operations are provided at Appendix J.

Requirements Identified. Not applicable.

Please use this space for your notes

2.4 Projects in the Planning Stage

The next step in the investigation was a study of projects in the planning stage. A number of those projects were in the process of defining requirements which could have an impact on how the new Automatic Scenario Generation System should be designed. Those projects are discussed in the following subparagraphs.

2.4.1 Battlefield Functional Areas System

General System Description. BFA is a collection of tactical communications systems designed for overall battlefield management. Several of the systems that are included in BFA are Advanced Field Artillery Data System (AFATDS), All-Source Analysis System (ASAS), and Maneuver Control System (MCS). ATCCS operates the various BFA area domains.

General Description of Scenario Generation. The messages tend to be relatively long text-oriented messages. Most immediate testing will probably be traffic oriented using representative dummy type messages. As testing continues, more message-oriented tests will be common.

Operational Details. Much of the information is location related, related to particular units or conditions at particular places. Map overlays and graphical entry of data are features of these systems. The systems are expected to be used in conjunction with test exercise (war-gaming) systems such as CBS/JESS in a free play environment.

Requirements Identified

Improved User Interface. A graphic scripting capability would be very helpful. Map overlays and graphic data entry would simplify the scripting process.

Support for Traffic-oriented Tests. Many of the early tests are likely to be traffic oriented. Later tests which are more message oriented may still require a background loading capability to simulate the loading stress of a battlefield environment.

TOELs. A simple TOEL capability is anticipated. Preliminary work on ATCCS made use of simple TOEL capabilities.

Interactive Scenario Execution. There is a desire for a free play capability for testing and training purposes. Interactive execution would be necessary to support that capability.

Responding to the SUT. We anticipate a requirement to respond to some types of messages received from the SUT.

Multi-node Scenarios. These systems involve a network with instrumentation at multiple locations.

Central/Remote Control. Preliminary work on ATCCS indicates a need for some basic remote control capabilities. We expect the TCC to be integrated to provide this capability.

Exercise Driver Interfaces. This could be useful, especially for training purposes.

2.4.2 Mobile Automated Instrumentation Suite System

General System Description. MAIS is a training/exercise system. It is intended to allow realistic battlefield-type training. It allows instrumentation to determine hit probabilities and track movement.

General Description of Scenario Generation. Instrumentation would be required at a number of points in the network. Some simulation capability may be required (e.g., simulate a tank). There is a lot of location-oriented data.

Operational Details. We anticipate that the tests will be message oriented rather than traffic oriented. A free play capability will be necessary. Currently MAIS testing is in the early planning stage. Probable instrumentation points have been identified but scenario requirements have not been established.

Requirements Identified

Improved User Interface. A graphical scripting capability would be useful for location data.

Interactive Scenario Execution. This feature will be necessary to allow flexible free play type testing and
training.

Simulation Capability. We believe that a simulation capability would be useful to allow the instrumentation to play various active roles.

Responding to the SUT. There will be requirements to respond appropriately to the SUT.

Multi-node Scenarios. We expect instrumentation at a number of points in the network. Multi-node scenarios would simplify the testing tasks.

Central/Remote Control. Central control capabilities would simplify test conduct.

2.4.3 MSE System for Option Year 4 Test

General System Description. MSE is a voice and data network communications system. The Pretest System for MSE is run on a combination of Micro-TISs and TISs. At the time this report was being developed, special requirements for the MSE were being defined for OY4 testing scheduled in September 1991. The basic MSE scenario generation process has been described in paragraph 2.2.3 above and in Appendix F. Discussed here is the unique way MSE will function for the OY4 test.

General Description of Scenario Generation. Starting in May 1991 an effort was made to improve the scenario generation process by generating a small scenario for MSE OY4. Up to that time, instrumented call scenarios were being produced manually (although part of the process was being automated). That process was thoroughly reviewed. The results of that review are provided at Appendix K.

Operational Details. See Appendix K.

Requirements Identified

Support for Traffic-oriented Tests. Messages are generated during real-time operations. The user can change the message generation rate.

TOELs. The system implements a simple, fixed TOEL capability. TOELs are predefined. This feature was added primarily for support of remote control capabilities.

Interactive Scenario Execution. A limited capability for changing/defining a scenario during real-time execution exists. The user can control the message rate and select TOELs.

Multi-node Scenarios. The old MSE TDAT Scenario Generation System is used to generate scenarios simultaneously for multiple nodes.

Central/Remote Control. The TCC is used to provide simple remote control capabilities. The user can load, hold, pause, or resume a scenario. The user can modify the message rate and request status information (message counts, etc.).

2.4.4 TADIL Analysis Tool System.

General System Description. The TAT system is a monitoring and analysis system designed for the Air Force. That system is also referred to as the Real-time Test Monitor (RTTM) system and will probably be run on a Sun work station and VME crate system.

General Description of Scenario Generation. Graphic scripting is required to the extent of allowing location-type information to be entered with a mouse. Users can script both messages and higher granularity type events. Scenarios can be created and modified during real-time execution.

Operational Details. The system requires a high degree of interaction with the SUT. It must respond correctly to messages received and can emulate a number of different tactical systems. An interactive scenario execution capability allows the user to step through the scenario and jump to different locations in the scenario. There are two separate scenario queues, one for manual and one for automatic operations.

Requirements Identified

Improved User Interface. Graphical scripting and windowing capabilities are a requirement. A graphical tactical display is required to show the current test status.

TOELs. TOELs could be a useful tool for implementing higher level objects.

Interactive Scenario Execution. Interactive control including single step, jumps, and manual control are required. The capability of creating new messages and events on-line during real-time execution is required.

Simulation Capability. The system is required to simulate the TADIL protocols. It also needs to emulate different Tactical Data Systems (TDS) implementations through support of individual Interface Design Handbooks (IDHs).

Response to the SUT. The system is highly interactive and must respond appropriately to messages received from the SUT.

2.4.5 Tactical Deployment Simulation System

General System Description. TADSS is a system intended to automate the labor intensive process of populating a tactical deployment with communication equipment based on a SCORES scenario. The system is being developed by a different contractor than the TIS contractor.

General Description of Scenario Generation. Refer to Appendix L.

Operational Details. Refer to Appendix J.

Requirements Identified. The TADSS system should be examined in more detail to determine if it would be useful to interface to this system to aid in scenario generation.

Please use this space for your notes

2.5 Identifying Requirements for New System

We approached the writeup of the various requirements identified for the Automated Real-time Test Scenario Generation System by establishing the general design requirements, determining appropriate methodologies, identifying feature enhancements, looking at potential problem areas, and evaluating the feature enhancements with regard to those problem areas.

2.5.1 Establishing General Design Requirements.

A number of design requirements have been identified. It is recommended that the following items be included in the Automated Real-time Test Scenario Generation System:

Maximized Reuse of Developed Software. Programming should be written to avoid system-specific code. Techniques such as table or data base driven software allow the program to be adapted to a new system without changing code. Ada provides features such as overloading and generic packages that can help in making the software more generic.

Off-the-Shelf Software. To save time and reduce costs, we should employ off-the-shelf software where possible. There are packages available for graphics management, prototyping, communications, data base management systems, etc. Some are independent software packages such as graphical user interfaces or data bases while others consist of libraries of subroutines. Ada is still somewhat deficient in this area. The design phase should examine the possibilities.

Real-time Integration. Considerable integration with real-time functions may be required. A decision should be made early in the design process on the degree of integration of scenario generation and real time.

• A VME-TIS Real-time System exists, and the cost of integration needs to be estimated. Many of the enhancements are by their nature real-time functions. The ALSYS kernel used by the Real-time System does not support graphics or an external communications interface such as Ethernet. The Real-time System could be enhanced to support Ethernet.

• Those features which require graphics or communication support may not be able to be integrated into the VME-TIS Real-time System. A two CPU system should be considered. A two CPU system requires communication between the Pretest Scenario Creation and Real-time Scenario Execution CPUs. An effective communications interface between the two systems will have to be supported.

Development Using CASE Tools

• CASE tools are required in the development of the Automated Real-time Test Scenario Generation System. CASE tools provide a central location for design information, which makes coordination and communication on the project simpler. Many CASE tools automate the task of providing documentation in accordance with DOD-STD-2167A.

• CASE tools make inconsistencies in the design specifications obvious early in the project. The earlier problems are found, the cheaper it is to fix them. CASE tools can provide a noticeable improvement in overall quality. There are several good CASE tools available for Ada.

• CASE tools can be expensive and it takes time to implement CASE methodologies, but the overall improvements in quality and productivity make it a worthwhile exchange.

• The CASE tool should allow several engineers to work on different portions of the design simultaneously. A product with networking support should be considered to support this capability.

• The CASE tool should support Ada code generation. The CASE tool should be able to generate code for Ada specifications and templates for Ada bodies.

• The CASE tool should support a standard integration framework. This would allow other tools to use the same specification data and would simplify possible future upgrades.

2.5.2 Determining Appropriate Methodologies.

An object oriented developmental methodology is required in the development of the Automated Real-time Test Scenario Generation System. Object oriented techniques should be used where possible. These methods provide a clean path for customization and tailoring of systems by developing objects of varying complexity as required.

2.5.3 Identifying Feature Enhancements.

A number of feature enhancements were identified, a feasibility analysis was then performed, and a final list of recommended feature enhancements was developed.

2.5.3.1 Possible Feature Enhancements.

The following feature enhancements were identified for the Automated Real-time Test Scenario Generation System:

Improved User Interface. The standard for user interfaces has advanced considerably since the TIS was originally designed. The current interface is not very user friendly and requires a high level of user expertise. Useability would be improved if we moved to a more graphical, pointer driven interface. Many scripting tasks would be better served by a graphical interface, e.g., track placement, network definition.

Support for Traffic-oriented Tests. We have already encountered several tests which required development of separate scenario scripting systems because of the lack of support for traffic-oriented tests. We need to support scripting in terms of loading in addition to the current simple lists.

• Table 2 illustrates the type of information that might be used to define a traffic-oriented scenario. The scenario defines message rate on the needline for each message type, and variance allowed in the rate and the type of distribution to be used on the messages. Note that a needline is a communications link between two nodes of a system or network.

• Table 3 illustrates the type of information that might be used to define a message-oriented scenario. The scenario defines the time each message is to be sent, the message type, and the content of the message.

Table 2. Traffic-Oriented Test Scenario

Needline 37			
Message Type	Message Rate (msgs per hour)	Variance	Distribution
msg x	100	0.00 %	even
msg y	100	5.00 %	random type 1
msg z	200	15.00 %	random type 2
TOTALS	400	8.75 %	

Table 3. Message-Oriented Test Scenario

Scenario Engagement		
Time	Message Type	Content
00:00	A	"START"
00:05	B	"UNKNOWN AIR TRACK #0015 AT HEX B12"
00:25	D	"DISPATCH F-15 #0017 TO HEX B7"
01:30	R	"TRACK #0015 AT HEX C14"
01:45	B	"TRACK #0015 EVALUATED HOSTILE"
01:50	D	"ENGAGE TRACK #0017, TARGET #0015"

Time Ordered Event Lists. A TOEL is basically a higher order of granularity for scenarios. The differences between our existing Pretest System and a new system using TOELs are described in the following subparagraphs.

• In our current system, a scenario is a list of timed events. An event is a list of timed messages. When the scenario is generated during pretest operations, the events are elaborated and the scenario file is simply a list of timed messages. Events are eliminated so far as the Real-time System is concerned. An illustration of the current system is provided in Figure 12.

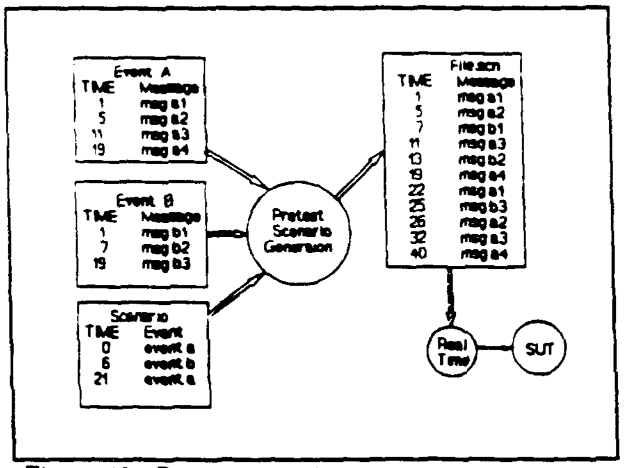


Figure 12. Pretest scenario generation using events.

• A TOEL is like an event in our current system. In a TOEL, the scenario could remain a list of events and the events would be elaborated by the Real-time System. A TOEL event could be a simple list of messages or a more complex system with parameters and conditional statements. AFATDS, MCS, and ATCCS would use this capability. TOELs would require close coordination with the Real-time System. An illustration of a system using TOELs is provided in Figure 13.

Interactive Scenario Execution.

• Operational testing is interactive and user driven. Users determine events on the fly based on the current situation. The current system has very limited capabilities for determining activity during real-time operations. We need to be able to compose scenarios and execute them during real time. We need more precise control over scenario execution in real time.

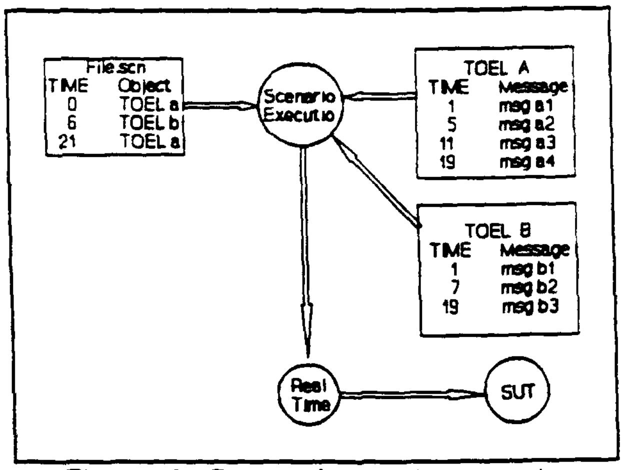
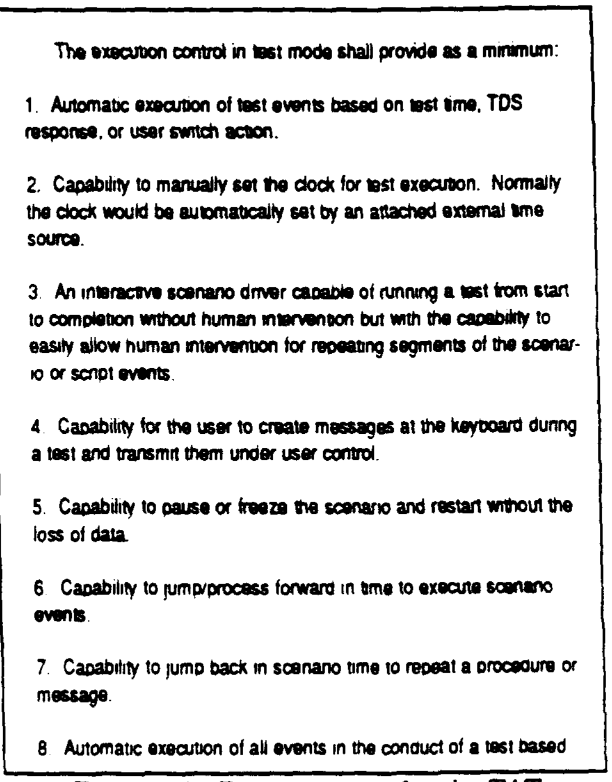


Figure 13. Proposed scenario generation using TOELs.

• The TAT system, for example, anticipates a very interactive environment with scenario execution controls such as single stepping, jumps, and manual control. The requirements for the TAT Interactive Execution capability from the SRS for that project are defined in Figure 14.



The execution control in test mode shall provide as a minimum:

1. Automatic execution of test events based on test time, TDS response, or user switch action.

2. Capability to manually set the clock for test execution. Normally the clock would be automatically set by an attached external time source.

3. An interactive scenario driver capable of running a test from start to completion without human intervention but with the capability to easily allow human intervention for repeating segments of the scenario or script events.

4. Capability for the user to create messages at the keyboard during a test and transmit them under user control.

5. Capability to pause or freeze the scenario and restart without the loss of data.

6. Capability to jump/process forward in time to execute scenario events.

7. Capability to jump back in scenario time to repeat a procedure or message.

8. Automatic execution of all events in the conduct of a test based

Figure 14. Requirements for the TAT interactive execution capability.

Simulation Support. Some systems require a simulation capability. Our JTC^{3}A TADIL-A/B/J systems currently have that capability to a limited extent. Simulation can be done to a number of levels. At the highest level the user would merely describe the conditions (environmental scripting) and the simulator would determine the message activity. At a lower level the user might script the original messages and the system would handle the repeats. Simulation (to any level) requires close coordination with the Real-time System.

Responding to the SUT. Often the instrumentation must respond to the SUT. The current JTC^{3}A TADILs are required to respond to the SUT during real time. The responses are provided by simulating parts of the TADIL protocols in the Real-time System. Requirements to respond to SUT activity during real-time execution are becoming more common.

Multi-node Scenarios. Many systems involve testing in a network environment with many nodes. Figure 15 shows an MSE network from the MSE OY4 Instrumentation Validation Test. Currently each instrumented node must have its own scenario created independently. We need to be able to create an integrated scenario involving multiple nodes. This would simplify the scripter's task by allowing easier support of interactions between nodes. It could allow more intelligent simulation by making activity on the network's other nodes visible to the simulator. At some point the instructions for each node would be broken out and supplied to the appropriate node's instrumentation.

Central/Remote Control Support. As stated in the preceding paragraph, many tests involve a multi-node network. Currently each instrumented node must have its own user. Some systems, e.g., ATCCS, need to have a control center which can operate all the instrumented nodes. This would allow better coordination of the test. It would also reduce costs by eliminating the users at each node. The TCC system currently under development is intended to provide this type of support. The communications interface between the TCC and scenario generation should be defined early in the design phase.

i. Exercise Driver Interfaces. There are several exercise drivers (war gaming systems) that we expect to see in use for testing. They may include CBS/JESS, DBITS, and TACSIM. Interfacing with these systems would allow the instrumentation to be aware of and to respond to the exercise environment. Such an interface would improve any modeling and simulation capabilities by improving the system's awareness of the environment. Such an interface would involve the cooperation of the exercise driver.

2.5.3.2 Potential Problems Areas. The following potential problem areas were evaluated in relationship to each proposed enhancement:

Real-Time Connection.

Coupling with Real-time. Most of our new requirements need a tighter coupling with the real-time portion of the system.

Existing Real-time Software. A VME-TIS Real-time System has already been developed. Some real-time SSAs will have been developed before design is started on scenario generation. It is difficult to change existing and working code to incorporate new design requirements.

Two CPU Approach. If changing existing VME-TIS code proves difficult, we could use the two CPU approach.

• The VME-TIS Real-time System would need to support communications with the Pretest Scenario Creation System. The Pretest Scenario Creation System would feed the data to be queued to the VME-TIS Real-time System for transmission to the SUT as it is generated. The VME-TIS Real-time System would send data seen on the link to the Pretest Scenario Creation System for analysis and action. The two systems are illustrated in Figure 16.

• One way to do this could be to use two separate boxes. The disadvantage of the two box system is a more complex system and development environment. Dividing the system into two parts is needed for both test conduct and development. A better possibility would be to have both "boxes" be cards or components in a single enclosure. To the user it would appear as a single system even though it would contain two different CPUs and operating environments.

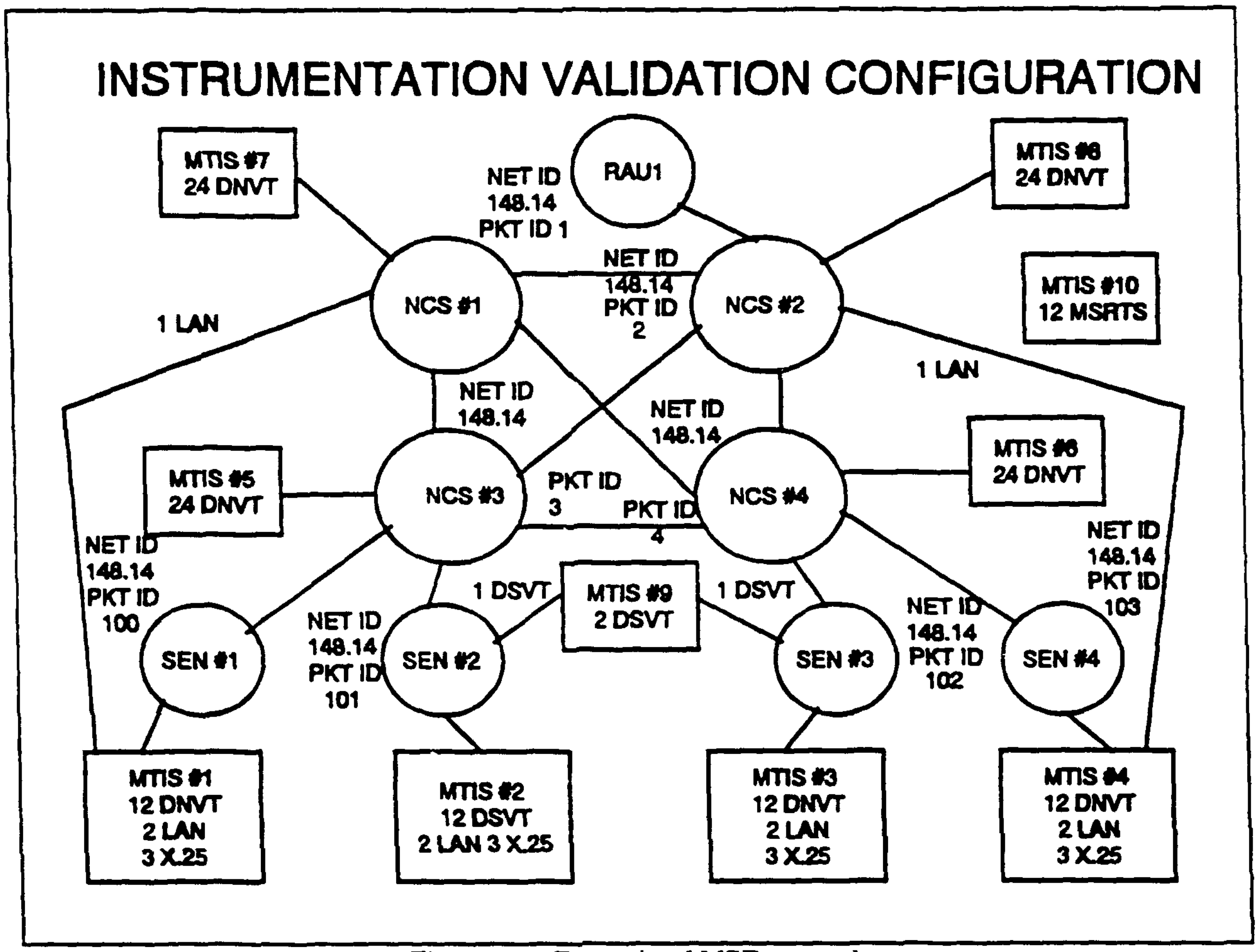


Figure 15. Example of MSE network.

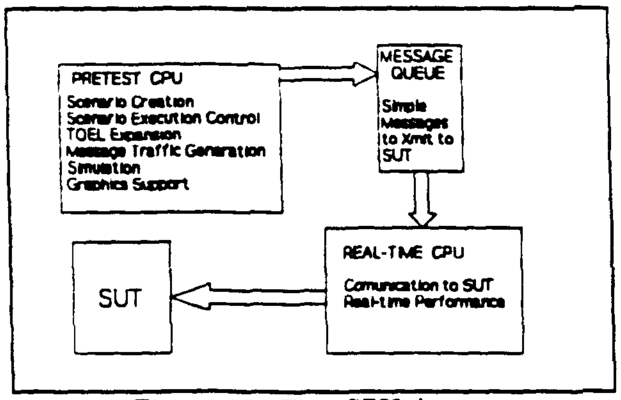


Figure 16. Two CPU design.

• On the positive side, the two CPU approach provides more CPU power by moving the pretest functions to the Pretest CPU. It might also provide better support for graphics.

System Specificity. Some items are by their nature specific to a single system. For example, simulation support would have to be developed for each system being simulated. Additional effort would be required for each new system. We need to use generic methodologies such as data base or table driven algorithms whenever we can.

Cost versus Benefit. Some systems will be involved in testing over a period of years. Other systems may be involved in tests that last only a few months. It may not be worth the effort to develop all features for every system. At times it may be better to have 50 scripters working with an unfriendly system than to invest in an expensive scripting system that will be used only once. We need to design a system that allows us to mix and match features using only what we need.

Tailoring. Even when cost is not a consideration, time may be. Sometimes we need something now and can do without a lot of features. At times important information needed to build some portions of the scripting system may not be available until shortly before the start of testing. We need a system that will allow system-specific details to be added quickly late in the development cycle or allow them to be left out completely.

2.5.3.3 Analysis of Potential Feature Enhancements. The potential feature enhancements were analyzed related to potential problems and consideration was given to possible implementation techniques.

Improved User Interface

Windows Environment. We are planning on a UNIX, X WINDOWS, environment for the new scenario scripting software. This will provide support for a graphics windows and mouse-type interface. Any new system can support an X WINDOWS look and feel for standard functions. Current standard functions include activities such as data entry, file selection and management, and command interface.

Graphical Interface. In addition to these traditional functions, we will have the opportunity to use graphical techniques for scenario definition. There are several possible levels of graphical interface integration. They are described in Figure 17. Figure 18 shows a screen for graphical scripting of track-type data. Figure 19 shows a sample screen for graphical scripting of traffic-type data.

Scenario Execution and Creation. Scenario exescenario. For example, the user might click on a scenario event or message to set a breakpoint (e.g., execute to this point and hold). The standard scenario creation features should be available (see paragraph 2.5.3.3.d, Interactive Scenario Execution).

Support for Traffic-oriented Tests

Real-time Support. The best approach to this problem is to provide support in real time for message traffic generation. An EPLRS scenario for a node might involve just one item, e.g., generate 400 type X messages per hour. This level of scripting would simplify the support for the Central/Remote Control feature. This kind of high-level scripting is also implicit in support for TOELs.

System Variations. Each Real-time System would need to support message generation for its type of messages. The basic system would not need to support every possible message type. The EPLRS system would have the EPLRS message generation modules linked in. The MSE system would link in MSE message generation, and so on. Systems which do not need any real-time message generation would not need to support it at all.

Object Oriented Approach. Using an object oriented methodology will be very beneficial. It would allow the Real-time System to deal with objects of varying complexity. Simple objects could be messages to be transmitted over the communications link (as in our current system). More complex objects might be traffic generation objects, TOELs, or even simulation objects. A given system would only need to support the objects it needed. Thus a simple system might only define message objects, which would save time and money. If a later test of the same system required more complex scenario support, more complex objects could be added to the existing system.

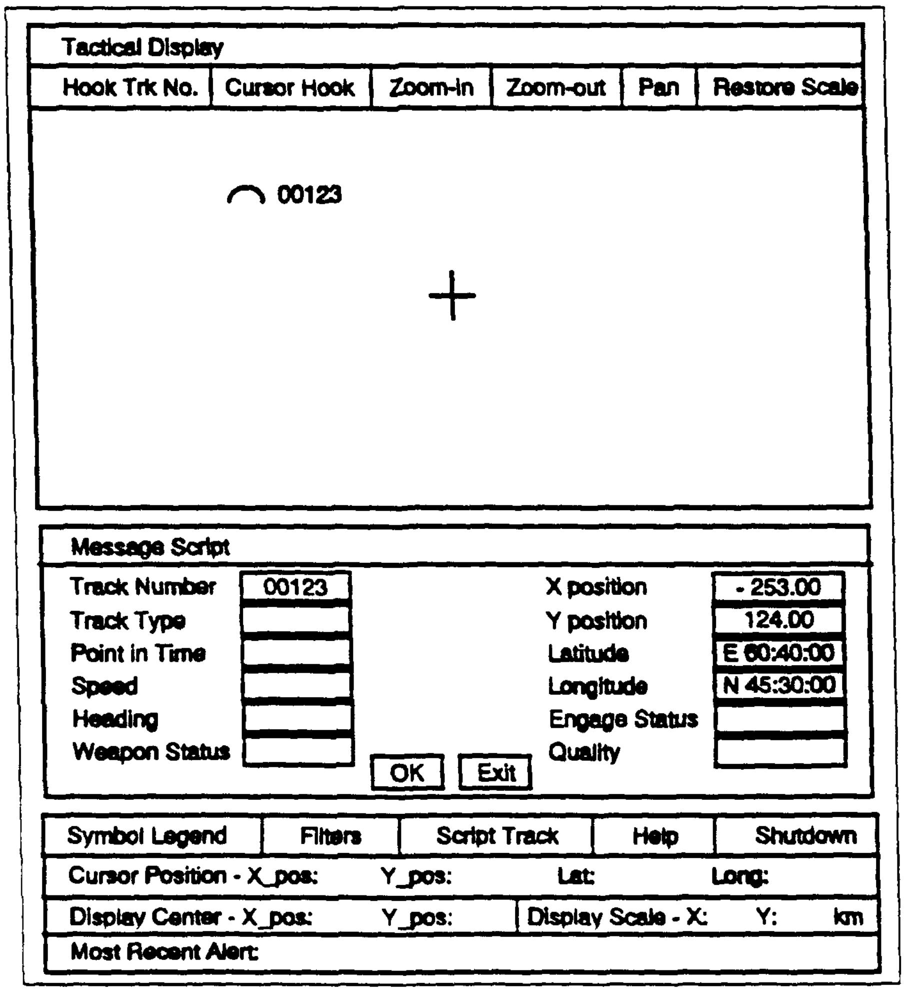
Use of Data Base or Tables. It may be possible to make traffic generation generic to some extent through the use of data base or table driven techniques. If the messages were described in a standard data format, perhaps new message types could be added to a data base without additional programming effort. Implementing a new system would involve creating its message generation data base. This possibility needs further study. Figure 20 shows a sample of the data tables used for TADIL message interpretation.

Time-ordered event lists

Object Oriented Approach. An object oriented development methodology would be useful. Scenarios would be a list of objects. One type of object could be a TOEL. TOELs could be simple lists of fixed objects (equivalent to events in our current system).

Level 1:	Single tracks and units can be placed on the screen using the mouse. In a traffic-oriented scenario the nodes could be placed by mouse and connections could be drawn between the nodes. The graphic feature is only used to define a single object.
Level 2:	The scenario definition can be displayed graphically to the user. The mouse can be used to create new objects or select existing objects for modification. In a traffic scenario the user could select a connection between nodes by mouse and define the traffic for it. Timing information would be entered manually.
Level 3:	The scenario state at any specific point in time could be displayed to the user if real-time functionality for scenario execution were integrated. The user could see the location of tracks or the state of traffic at N minutes into the scenario and modify or add to the scenario based on its "current" state.

Figure 17. Graphical interface levels.



Track Number	00123	X position	- 253.00
Track Type		Y position	124.00
Point in Time		Latitude	E 60:40:00
Speed		Longitude	N 45:30:00
Heading		Engage Status	
Weapon Status		Quality	

Cursor Position - X_pos: Y_pos: Lat: Long:

Display Center - X_pos: Y_pos: Display Scale - X: Y: km

Figure 18. Track-type graphical scripting screen.

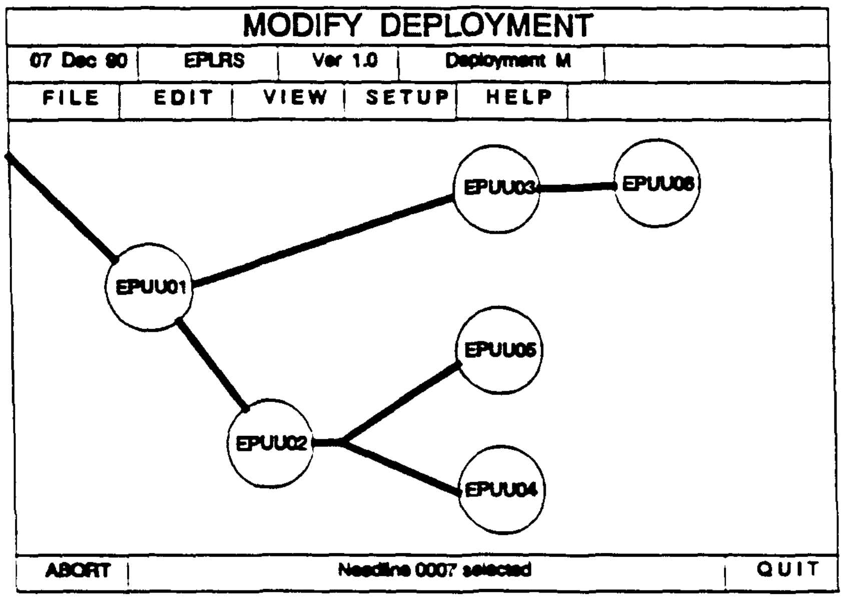


Figure 19. Traffic-type graphical scripting screen.

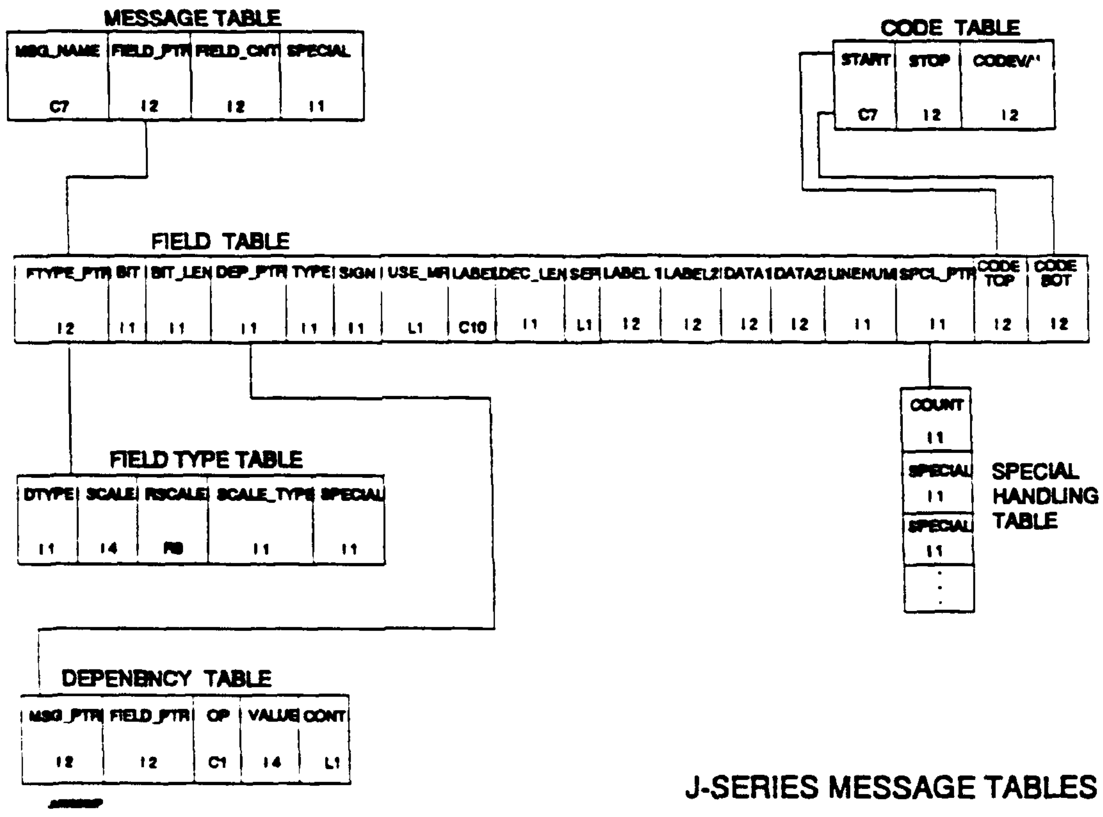


Figure 20. Data tables for TADIL message interpretation.

Elaborating a TOEL. When the Scenario Execution capability in the Real-time System encounters a TOEL, it would simply look up the list and send it. This would be equivalent to elaborating events in real time.

Central/Remote Control. TOELs simplify support for the Central/Remote Control Support capability.

Real-time Support. The Scenario Execution capability of the Real-time System would have to add support for this feature but the effort should be relatively small.

Expansion of TOELs. TOELs could be expanded in several useful ways. They can be thought of as the scenario's equivalent of subroutines. TOELs could allow passed parameters. This would allow a single TOEL to be used in different circumstances, e.g., to specify a track ID as a parameter to a drop track TOEL. Control information could be added to the TOEL, e.g., if traffic is greater than 400 messages an hour, use random distribution, otherwise use equal spacing.

Advanced TOELs

• Advanced TOELs add parameters and control capabilities to the standard TOEL capabilities. Tables 4 and 5 show the type of data that might be used to define an advanced TOEL.

•When the scenario invokes the TOEL, it will include the values for P1 and P2 and those values will be put into the message contents during real-time execution. The control feature skips messages based on the values of input parameters. The first invocation of the TOEL skips the message four because P1 is greater than 800. The second invocation of the TOEL will skip messages three through five because P2 is greater than 3 (see the condition on message two).

Table 4. Definition for TOEL Engagement

SEND TIME	MSG TYPE	MESSAGE CONTENTS				CONDITION	ACTION
		FLD 1	FLD 2	FLD 3	FLD 4		
00:01	A	127	15	7833	789	None	
00:06	D	876	P1	8	P2	P2 > 3	Jump forward 4
00:12	M	1200	150	666	615	None	
00:20	A	4	77	P2	374	P1 > 800	Exclude
00:20	K	225	978	34	1928	P1 <= 800	Exclude
00:20	R	1830	1847	1954	14	None	

Table 5. Scenario File

TIME	OBJECT	PARAMETERS			
		P1	P2	P3	P4
00:00	TOEL ENGAGEMENT	1100	2		
01:00	TOEL ENGAGEMENT	630	7		

Generic TOELs. Basic TOELs could easily be made generic. The enhanced TOELs could be generic if we can come up with a data-driven methodology. Again, further study would be needed to determine the practicality of generic enhanced TOELs.

Interactive Scenario Execution

Scenario Creation / Scenario Execution. We have generally avoided the term "Pretest" in this paper because it implies the traditional TIS-type system with independent Pretest, Real-time, and Posttest Systems. Pretest actually involves two capabilities, scenario creation (definition) and scenario execution. Scenario execution under the TIS has been an automated process in real time. We now need a Scenario Creation capability available during real time and interactive user control over the Scenario Execution capability.

Current Commands. Currently the user's control of a scenario in real time is limited to a few basic commands (start, stop, pause, hold).

Interactive Capabilities. In the more interactive environment of many tests, the user wants complete control of the activity. Some of the capabilities we will need to support those tests are included in Figure 21.

Central / Remote Control. These features should also support the Central/Remote Control capability. A user at the central control station will want to be able to control the Scenario Execution capability at different nodes during real-time execution.

Posttest Analysis / Test Repeatability. Interactive scenario execution complicates posttest analysis and test repeatability. The test log file will have to contain scenario execution activity to allow the analyst to determine what should have happened. It is no longer possible to simply check the scenario file since it may bear little relation to what actually occurred during execution. It would be useful for test repeatability to be able to replay scenario activity exactly as it happened in a previous test. This would help in version testing and analysis.

Graphics Environment

- We anticipate using a graphics/mouse-type environment for scenario creation. This will not pose a problem in a UNIX, X WINDOWS environment.
- The current VME-TIS runs under a different environment. The VME Real-time System uses the ALSYS real-time kernel which has no graphics support. A third party add on (ADA

Manual Control:	Features similar to a source code dynamic debugger. Single step, set breakpoints, jump to specified location (forward or backward).
Automatic Control:	Change speed from scripted values, (half speed, double speed, etc.). Skip marked portions of a scenario or send marked portions to separate queue for manual execution.
On-line Creation:	Create messages or events during real time and execute them. The user will expect the same scenario creation features that were available for prescripting scenarios (graphics, etc.).
Environment:	Modify existing objects in a scenario. For example, in a traffic-type scenario the user may want to change the message rate from 400 messages an hour to 600. A user may want to change an existing track's identity from unknown to friend. Ideally the user would make his/her change, and the Real-time System would generate the message traffic. Environmental modifications are an extension of on-line creation.

Figure 21. Interactive capabilities.

WINDOWS) supports sufficient character-based graphics to do interactive scenario execution (breakpoints, etc.).

• The current VME-TIS Real-time System could not support a good tactical display or graphical scripting. This could be solved with a separate work station handling the graphics and communicating with the Real-time System as described in paragraph 2.5.3.2(a)(3). Support would have to be added to the VME-TIS Real-time System for the communications interface

Simulation Support.

Ease of Scripting. Full simulation support is the ideal in ease of scripting. It allows a user to concern himself/herself with what is happening and ignore how it is happening. It allows users who are not experts to do most of the actual scripting work. We currently use simulation capabilities in some of our SSAs to one degree or another.

Problems. There are problems with providing this level of support as described in Figure 22.

Advantages. In spite of the potential difficulties of simulation support, it can be a very useful tool and is necessary in some circumstances. It would simplify support of the Central/Remote Control capability by reducing message traffic. The Interactive Scenario Execution capability would be simplified as well and would easily allow environmental-type changes.

Case-by-Case Analysis. Simulation should be handled on a case-by-case basis with careful cost/benefit analysis done before development.

Real-time Integration. The simulation model must be executed in real time. The VME-TIS Real-time System would need the power to do its normal tasks and maintain the simulation. Simulation would be tightly integrated with real-time operations. A two CPU solution could be used if the real-time processor does not have necessary power.

Object Oriented Methodology. An object oriented development approach would be helpful. It allows creation of objects of varying complexity according to the needs of the test. For example, a message X object could be created which models the protocol for message X. The message Z object might know nothing of protocol. On a higher level, a track object could be developed which simulates the behavior of a track. If inheritance were available, the track object might be extended later to include motion modeling (e.g., fly a race track or circle pattern). Complexity could be included or ignored based on the needs of the test and cost considerations.

Responding to the SUT

Relationship to Other Features. Some of the other feature enhancements would allow the system to respond to the SUT during real-time.

• Interactive Scenario Execution. This feature will allow the user to create messages in real time. The user can compose messages based on traffic seen from the SUT. The user can select prescripted events or messages to send. This allows anticipated actions to be scripted in advance for rapid responses during real time. Sometimes the SUT requires a response that is too rapid to allow for human interaction. Some systems require responses on the order of milliseconds. Interactive scenario execution would not support those requirements. Interactive responses to the SUT also require the constant attention of the user.

• Simulation. This feature can allow the system to respond to the SUT as long as the data from the SUT is used to update the current model's status. This is the method used by the JTC^{3}A TADILs. It allows responses to be generated without the intervention of the user. It permits response times as rapid as the hardware and software will support. It is very flexible since the simulation capability is developed for each system individually. The simulation method shares the general simulation disadvantages of high cost and applicability to a single system.

Response Table Approach

• A more generic approach is the use of response tables. This method would use a data table defining possible responses and a trigger mechanism which activates a response based on input from the SUT.

• The response table entry would look like a scenario or TOEL. It could contain commands or parameters for a Message Traffic Generation feature in a traffic-oriented system. The trigger mechanism could be something along the lines of

Cost:	It can be very expensive to provide accurate simulation to any level of detail. For example, we may need to know the specification of every sensor system that might be involved in order to determine which sensors would report which tracks with what track quality. We might also need to know the radar cross sections of different platforms in different aspects. For example, we detect a B52 at 200 miles in profile but at 175 miles when head on, an F-16 is detected at 125 miles, etc. We can do simulations to various levels of detail. For example, we may model radar detection but require the user to handle platform performance manually. A cost/benefit analysis would need to be done for each system.
Specific:	Many aspects of simulation do not carry over from system to system. The behavior of a TADIL-A system will be different from a TADIL-J system and so on. Many aspects of a simulation will have to be developed for each system to be tested.
Upkeep:	The effort involved in keeping a simulation up to date can be considerable. The real world changes quite rapidly and an out-of-date model is often worse than no model at all.
Errors:	Any errors in the model will affect the entire test. Errors may not be obvious until detailed analysis is done, resulting in repeated tests. Real world models are complex and error prone.
Control:	The user loses a level of control to the simulation's model. If the simulation determines that when a message X is sent, a message Y should be sent automatically, the user loses exact control. There may also be circumstances (e.g., in error testing, to check if the SUT will recover properly if no Y is received) where the user may want to violate the model.

Figure 22. Problems with simulation.

the event template as designed for RTTM. It would match patterns like "A message X with a type field of subsurface, followed within 15 seconds by a message Z with a depth field greater than 200." When the pattern is matched, the associated response (TOEL, message generation, or simulation command) would be sent. An example of a response table is provided at Figure 23.

• The response table method allows the rapid response times of simulation in a generic manner. The cost would be that of the original development. The only system-specific costs would be to define the responses and triggers for the system.

• This capability could be sufficient for many systems which might otherwise require a simulation capability. This method might also provide an easy path to handle exercise driver interfaces. CBS updates might be sent to the trigger mechanism as well as SUT link data. The CBS data, or a combination of CBS and SUT data, could trigger a response. Figure 24 describes a possible system configuration.

Multi-node Scenarios.

Node Address. Multi-node scenarios could be handled by adding a node address in the scenario. Where multiple nodes are not needed, the node address could default to some standard value. Actual nodes could be mapped to a node address either before or during the test. The scenarios could be processed to generate a separate scenario for each node and then be distributed by network. It would also be possible for the complete scenario to be executed by each node. The scenario execution process would simply skip commands addressed to other nodes. In a central/remote control environment, individual scenario commands might be distributed to the individual nodes during real time.

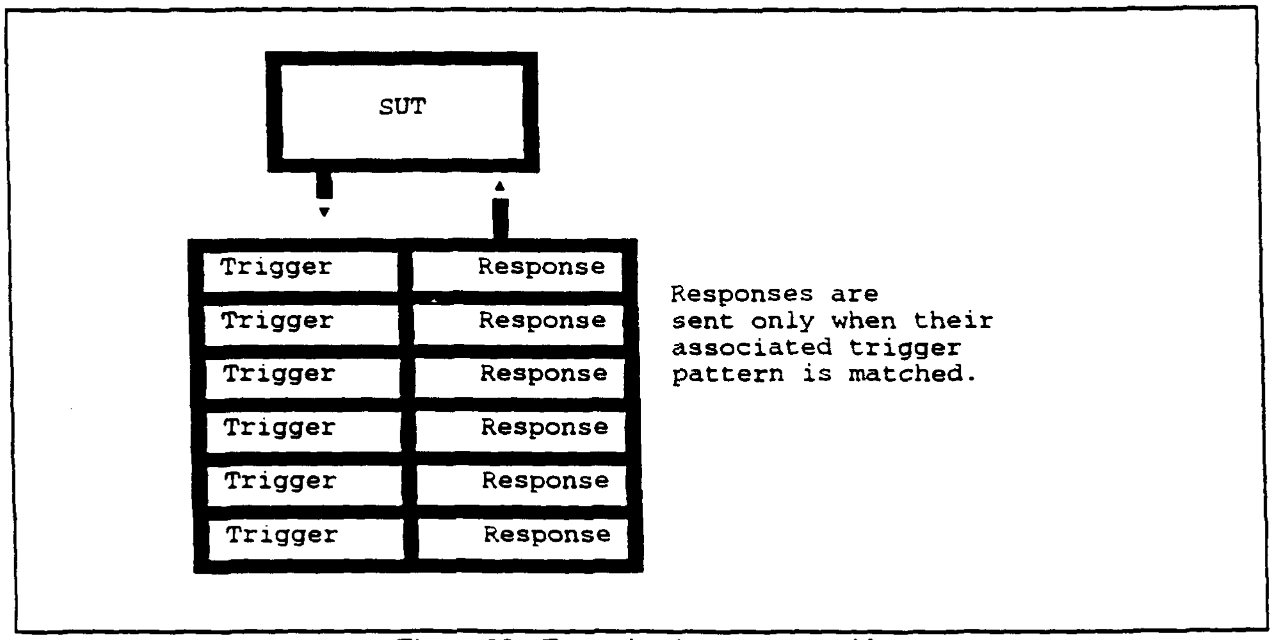


Figure 23. Example of a response table.

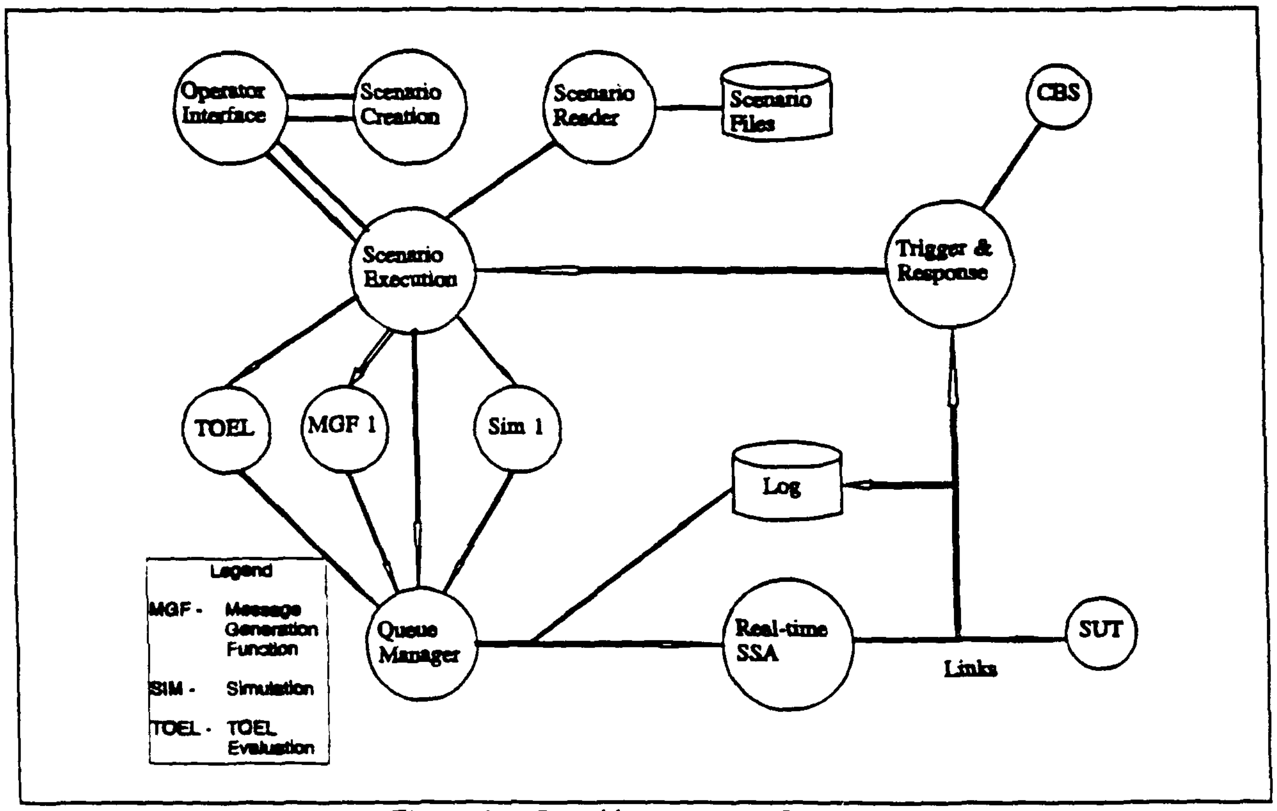


Figure 24. Possible system configuration.

Feasibility. It should be possible to have support for multi-node scenarios work in a generic manner. The effort required should be moderate. Taking full advantage of the additional possibilities such as better analysis, simulation, and network awareness will require more time and effort.

Central/Remote Control Support.

The TCC. The TCC is designed to provide central and remote control capabilities to a test. The MSE OY4 test in September 1991 will use the TCC to perform basic control functions on remote TIS instrumentation. Use of the TCC should be included in the design of the Automated Real-time Test Scenario Generation System. As part of the early design phase, the communications interface should be clearly defined. An extensible message set should be designed to allow new functions to be easily added to the remote capability.

Relationship to Other Features. Several of the previously discussed features will make central/remote control an easier task.

- The communication link between the remote instrumentation and the central control may have a low throughput. It will be necessary to minimize the traffic between the central station and its remote nodes. Traffic-oriented tests should work well in this environment since the scenarios are simple and short.
- TOELs allow a single object sent to the remote node to be expanded into many messages. The TOEL definitions should be distributed to the remote node before the start of the test to save time.
- Any significant simulation capability also reduces the message traffic. Instead of sending multiple messages to change a unit's status, a single message would do. For example, the central node might request the current ID of a track, and then change that single field.

Levels of Support.

- The simplest support would involve one way communications to the remote node. Commands would simply tell the scenario execution process to run scenario X or send this message/TOEL.
- More complete support would allow two way communications. The central control could query the remote node about its current state.
- Complete support would allow full interactive scenario execution as described in paragraph 2.5.3.3.d. The central control operator might have several windows open, each controlling a different remote node. The central control could see the network from any instrumented remote node's point of view.

Exercise Driver Interfaces.

Purpose of Interfaces. This area needs further study. It is not clear yet what the user will want to accomplish with these interfaces. There is a great deal of potential in this area, particularly for systems with some simulation capability.

Test System Interface. We will need to study exactly what sort of communications the different exercise interfaces support. Since the exercise drivers may have been designed without consideration of a possible test system interface, a development effort may be required on the part of those developers as well as on ours.

Interface with CBS. The CBS currently provides a generic interface, the JESS Generic Interface (JGI) and JESS Interface Toolbox (JIT), which allows an application to be informed of changes to the CBS data base. It may be possible to acquire additional CBS data through file transfers over the network without disturbing the current system. We could use activity seen on the CBS as a trigger to set off TOELs, change message traffic generation activity, or modify the simulation environment data base. Information on the CBS Interface, including diagrams of the CBS and TOEL command interfaces and the JIT/AUTOMsgs Data Flow, is provided at Appendix D. Figure 25 illustrates the CBS system configuration.

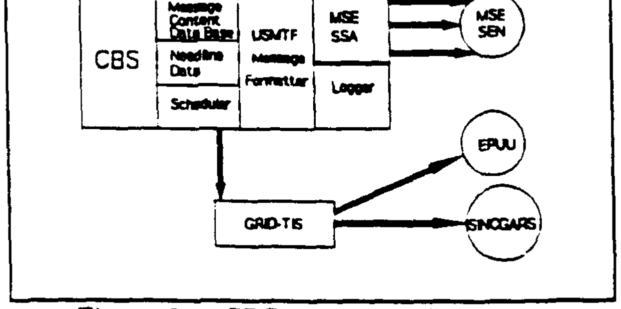


Figure 25. CBS system configuration.

2.5.3.4 Summary of Feature Enhancements Related to Problem Areas.

Table 6 summarizes each of the feature enhancements discussed previously. Each enhancement is analyzed in relation to the general problem areas described in paragraph 2.5.3.2.

- The Real-time Support column indicates the degree of effort which will be required on the part of the Real-time System.
- The System-specific column indicates whether the effort must be repeated for each system supporting the feature.
- The Cost column shows the overall effort required to implement the feature.
- The Tailor column indicates whether the effort can be tailored based on the individual system's requirements.

Table 6. Analysis Summary Table

Feature	Real-time Support	System-specific	Cost	Tailor	Notes
Improved User Interface	No	No	Moderate	Small	Windowing/mouse environment
Graphical Scripting	Yes	Yes	Moderate to High	High	3 levels of support possible; need graphics support in Real-time System
Support for Traffic- oriented Tests	Moderate	Yes	Moderate	High	Create object for each traffic generation type
Basic TOELs	Moderate	Data Only	Small	High	Add TOEL reading to scenario execution, create data for each TOEL
Advanced TOELs	Moderate	Data Only	Moderate	High	Include passed parameters and control flow
Inter-active Scenario Execution	Major	No	Moderate	N/A	Create new interactive scenario execution process
Simulation Support	Major	Yes	Moderate to High	High	Create object for every item to be simulated
Response to SUT	Major	Data Only	Moderate to High	High	Add response table and trigger system
Multi-node Scenarios	Minor	No	Small	Use/ Don't Use	Add node addressing to scenario creation and execution
Central/Remote Control Support	Major	No	Moderate	Use/ Don't Use	Add communication support and command structure
Exercise Driver Interfaces	Major	Yes; possibly data driven	High	High	Add communications and intelligence to make use of data; could use existing features like TOELs, simulation, and interactive execution capabilities

2.5.3.5 Recommended Capabilities. The final recommendations on the proposed feature enhancements are in the following subparagraphs:

Graphic Interface

• We should include the basic windows/mouse/graphics type of interface as part of the basic design. The VME-TIS Posttest System currently uses this sort of interface. It is relatively inexpensive to include this as part of the basic design. Some possible difficulties are discussed under paragraph 2.5.3.5.d, Interactive Scenario Execution.

• This capability is associated with the improved user interface feature shown in Figure 26.

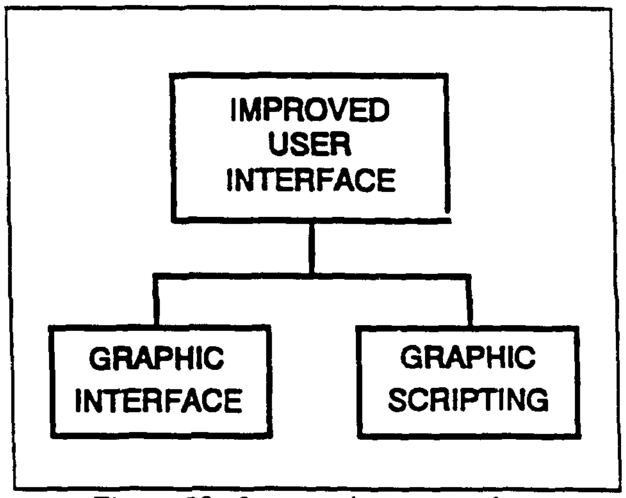


Figure 26. Improved user interface.

Graphic Scripting

• Each system should be analyzed to determine the level of graphic scripting to be supported. Cost versus benefit should be considered. It is possible to support both traffic- and message-oriented systems effectively.

• A graphics front end product called DataViews is available from V.I. Corporation. It provides Ada bindings, real-time control/display, rapid prototyping, and image importing. It supports X WINDOWS and runs on UNIX and VMS platforms. It might be possible to develop full Level 3 graphical scripting in a cost effective way using this or a similar tool. (Level 3 scripting is defined in Figure 27.) Such a tool could also be used to make the basic interface more flexible and user friendly.

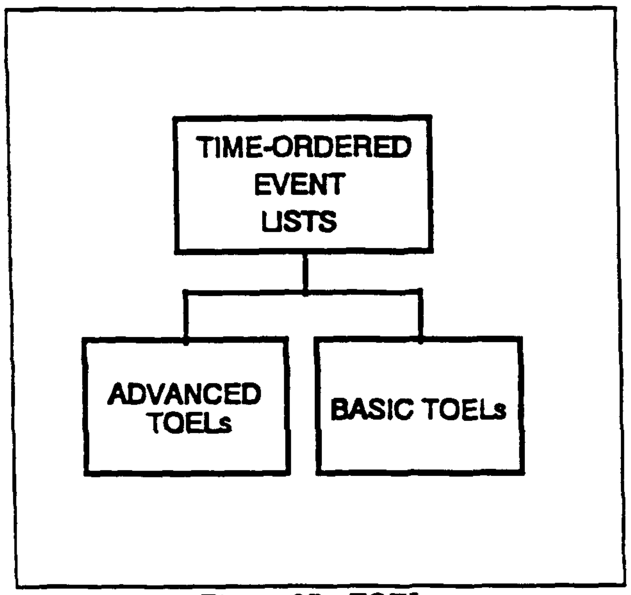


Figure 27. TOELs.

• This capability is associated with the improved user interface feature as shown in Figure 26.

Message Traffic Generation. Hooks for message traffic generation capabilities should be provided as part of the basic design. The actual traffic generation capabilities functions should be developed as part of the system-specific development efforts. The traffic generation capabilities would only be linked into the Pretest Scenario Creation System for a specific system. Traffic generation is a real-time capability. The current VME-TIS Real-time System can be enhanced to support traffic generation.

Basic TOELs

• Basic TOEL support should be included in the design. TOEL definition should be added to the basic Pretest Scenario Creation System. TOEL elaboration should be added to the basic VME-TIS Real-time System. The current VME-TIS Real-time System can be enhanced to support TOEL elaboration.

• This capability is associated with the TOELs feature as shown in Figure 27.

Advanced TOELS

• Further design work should be done to determine whether advanced TOEL capabilities (parameters, control flow, etc.) are desired and cost effective. If we decide to do advanced TOELs, they should be supported in the original design. The current VME-TIS Real-time System can be enhanced to support advanced TOEL elaboration.

• This capability is associated with the TOELs feature as shown in Figure 27.

NOTE: TOEL support is a one-time development cost. Once TOEL support (either basic or advanced) has been added, the only cost per system is the actual scripting of the TOELs used in the test.

Manual Interactive Execution

• The VME-TIS Real-time System should be enhanced to support manual control. Single steps, breakpoints, and jumps should be supported. The VME-TIS Real-time System cannot support a full graphical interface, but the current character-based windowing system is adequate for this capability.

• The VME-TIS Real-time System should be enhanced to support logging of scenario execution. A capability should be developed to extract the scenario execution activity from a log file into a scenario file for automatic re-execution. This is a requirement to support any of the interactive scenario execution functions.

• This capability is associated with the interactive scenario execution feature as shown in Figure 28.

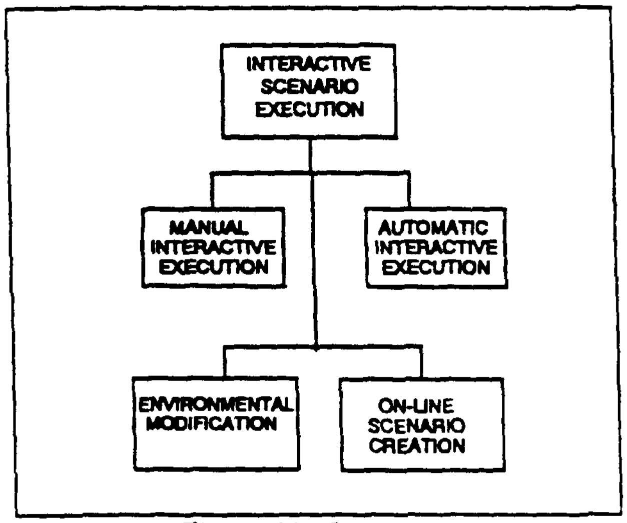

INTERACTIVE SCENARIO EXECUTION
MANUAL INTERACTIVE EXECUTION
AUTOMATIC INTERACTIVE EXECUTION
ENVIRONMENTAL MODIFICATION
ON-LINE SCENARIO CREATION

Figure 28. Interactive scenario execution interface.

Automatic Interactive Execution

• This capability should be examined in more detail. Hooks to support automatic controls should be included in the design. Implementation could be delayed until a test with these requirements needs to be supported.

• This capability is associated with the interactive scenario execution feature as shown in Figure 28.

On-line Scenario Creation

• This capability presents a problem. The VME-TIS Real-time System does not support a graphics interface. A graphical scripting system could not be supported. Operating environments which are capable of supporting real-time response requirements do not support good graphics.

• A possible solution is to run the Scenario Creation function in one environment to create scenarios and transfer the data to the VME-TIS Real-time System for execution. The ALSYS kernel does not support a standard communications interface such as Ethernet, but could be enhanced to support it.

• This capability will be a requirement for some tests. We should include hooks for support of this feature in our scenario generation design.

• This capability is associated with the interactive scenario execution feature as shown in Figure 28.

Environment Modification

• Those systems which support simulation and message generation features should support the ability to modify the environment during real-time execution. This capability is not applicable to simpler systems which do not generate messages in real time. The capability to modify the environment should be developed as an integral part of the Message Traffic Generation or Simulation capability.

• This capability is associated with the interactive scenario execution feature as shown in Figure 28.

Simulation. A cost/benefit analysis should be done to determine the degree of simulation support to be provided for each test. The VME-TIS Real-time System should be enhanced to allow hooks or sockets for simulation routines. Simulation capabilities should be developed on a case-by-case basis as the need arises.

Response Table

• A generic capability for responding to real-time SUT data should be developed. A trigger /response table system should be designed. The response table should allow any command or message which can be scripted to be included in the response. The trigger mechanism should be flexible. It should not be limited to looking at a single item. It should allow field values and times to be criteria. If the first tests to be supported do not require real-time responses, hooks could be provided and the actual development postponed until needed.

• This capability is associated with the responding to the SUT feature.

Multi-node Addressing. The Pretest Scenario Creation system should include support for node addressing. The Real-time Scenario Execution capability should be enhanced to be aware of node addresses and ignore objects not addressed to its node.

Central/Remote Control Support. Central and remote control support is simplified by support for the Message Traffic Generation, Basic TOELs, Advanced TOELs, Simulation, and Multi-node Addressing capabilities. If we support these features, the remaining problems are in the real-time domain of communications and interactive execution. The interface to the TCC should be clearly defined along with the messsage/command set. The interface could be implemented in Phase I.

Exercise Driver Interfaces. Further requirements analysis should be done in this area. Interfacing capabilities of the different exercise drivers should be studied as a part of the design effort for the Automated Scenario Generation System. Our goal should be to avoid closing off any interfacing possibilities. The CBS/JESS system supports communications which we could use to monitor its activity. It will be a major effort and should be implemented as needed.

2.5.3.6 Summary of Recommendations

Table 7 summarizes the recommendations regarding the feature enhancements discussed previously.

Table 7. Recommendation Summary Table

Item	Enhanced Module	When to Develop	Comments
Graphic Interface	Pretest Scenario Creation	Phase I	Part of hardware and software upgrade; consider on-line scenario creation requirements in design
Graphic Scripting	Pretest Scenario Creation	Hooks in Phase I, develop as needed	System specific; may not be difficult if off-the-shelf tools are available; consider on-line scenario creation requirements in design
Message Traffic Generation	Real-time SSA	Hooks in Phase I, develop as needed	Hooks same as simulation capability; development is system specific
Basic TOELs	Real-time Scenario Execution	Phase I	Not difficult; is a generic feature
Advanced TOELs	Real-time Scenario Execution	Further study required; if desired, do hooks in Phase I, develop as needed	No immediate need but could be useful

Table 7. Recommendation Summary Table

Item	Enhanced Module	When to Develop	Comments
Manual Interactive Execution	Real-time Scenario Execution	Phase I	Not too difficult, generic, and compatible with current real-time capabilities
Automatic Interactive Execution	Real-time Scenario Execution	Hooks in Phase I, develop as needed	Probably not needed immediately
On-line Scenario Creation	Pretest Scenario Creation Real-time Scenario Execution	Phase I if possible	Strong need but requires real-time graphics capabilities; consider two CPU design
Environment Modifica-tion	Real-time SSA	Develop as part of Message Traffic Generation and Simulation capabilities	Specific to simulation-type activities
Simulation	Real-time SSA	Hooks in Phase I, develop as needed	System specific
Response Table	Real-time SSA	Hooks for generic Response Table capability in Phase I; develop when needed	Generic capabilities should be developed in Phase I if budget permits
Multi-node Addressing	Pretest Scenario Creation Real-time Scenario Execution	Phase I	Simple and generic
Central/ Remote Control Support	Real-time Scenario Execution	Necessary hooks are already included in other capabilities; develop as needed	New capability for real time; is generic capability; provide TCC interface
Exercise Driver Interfaces	Real-time SSA	Additional study in Phase I; develop as needed	Consider in Phase I design to avoid closing possibilities; CBS/JESS looks promising

Please use this space for your notes

Section 3. Appendices

Please use this space for your notes

Appendix A. Methodology Investigation Proposal

1. Scope

This appendix contains the Methodology Investigation Proposal on which this methodology investigation report was based.

2. Proposal

FY92

Test Center: EPG

Mission Area
Supported: COM
Priority: 6

FUNDING $(K)

FY91	FY92	FY93	FY94	FY95	FY96	FY97	FY98
50	F50	50					

Title: Automated Real-Time Test Scenario Generation II

Principle Investigator: Curtis C. Massie
Email Address: NA

Office: STEEP-TS-C
Autovon: 821-8176

BACKGROUND: The Micro Test Item Stimulator (Micro-TIS) is a test driver which provides messages to a system under test (SUT). The test driver currently responds to incoming messages from the SUT with prescribed responses. A method of automated interaction with the SUT is required.

PROBLEM: Changing the Micro-TIS test driver's prescribed responses is an untimely process. The test driver does not have the capability to respond to a message in the same manner as the SUT responds. The test driver cannot participate in the interactive exercises except as noted above.

OBJECTIVE: Determine a method for automated interaction of the Micro-TIS test driver with the SUT. Phase I will define the required automated scenario generation system. Phase II will create a prototype of that system and develop interfaces so it can be used real-time to respond to a message sent from the SUT.

IMPACT/JUST: The current Micro-TIS test driver has limited capability to participate interactively with the SUT on a real-time basis. This capability is especially needed on systems with human operators in the loop, and on some systems just to keep the link operational.

PROCEDURES:

Target Date	Achievement
04/04/91	Complete design of automated scenario generation system.

07/10/91	Complete first methodolcgy report.
01/02/92	Build prototype of automated scenario generation system.
07/10/92	Complete interfaces so the automated scenario generation system can be used real-time to respond to messages sent from the system under test.
09/01/92	Complete second methodology report.

COST CATEGORIES:

■ Personnel Compensation:	$25K
■ Travel:	$ 0K
■ Contractual Support:	$25K
■ Consultants & Services:	$ 0K
■ Materials & Supplies:	$ 0K
■ Equipment:	$ 0K
■ General & Admin costs:	$ 0K

■ Man-hours Required:

■ IN-HOUSE DIRECT LABOR:	800
■ CONTRACT LABOR:	500

OBLIGATION PLAN:

obligation rate (Thousands)	FQ	1	2	3	4	TOTAL
		10.0	20.0	10.0	10.0	$50K

Appendix B. Acronyms and Abbreviations

AFATDS	Advance Field Artillery Data System
AI	Artificial Intelligence
ASAS	All Source Analysis System
ASCII	American Standard Code for Information Interchange
ATCCS	Army Tactical Command and Control System
AUTOMsgs	Automatic U.S. Message Text Format Test Output Messages
BFA	Battlefield Functional Areas
C^3I	Command, Control, Communications, and Intelligence
CASE	Computer Aided Software Engineering
CBS	Corps Battle Simulator
CEWI	Combat Electronic Warfare Intelligence
Comm	Communications
CP	Co-processor
CPU	Central Processing Unit
CSC	Computer Software Component
DBITS	Deep Battle Integration Test Support
DOS	Disk Operating System
DRAM	Direct Random Access Memory
EFL	Event Format Library
ENB	Engineering Notebook
EPLRS	Enhanced Position Location Reporting System
FY	Fiscal Year
Grid-TIS	Grid-Test Item Stimulator
ID	Identification
IDH	Interface Design Handbook
I/O	Input/Output
IOC	Initial Operating Capability
JDRA	JTIDS Data Reduction Applique
JESS	Joint Expert Support System
JGI	JESS Generic Interface
JIT	JESS Interface Toolbox
JTC^3A	Joint Tactical Command, Control, and Communications Agency
JTIDS	Joint Tactical Information Distribution System
Kbits	Kilobits
MAIS	Mobile Automated Instrumentation Suite
MB	Megabytes
MCS	Maneuver Control System
Micro-TIS	Micro-Test Item Stimulator
MIP	Methodology Investigation Proposal

MIPS	Millions of Instructions per Second
MSE	Mobile Subscriber Equipment
Msg	Message
MSRT	Mobile Subscriber Radio Telephone Terminal
OY4	Option Year 4
PTT	Portable TADIL Tester
RTTM	Real-time Test Monitor
SEN	Small Extension Node
SSA	System-specific Applique
SSAP	System-specific Applique Processor
SUT	System Under Test
TACSIM	Tactical Simulation
TADIL	Tactical Digital Information Link
TADSS	Tactical Deployment Simulation System
TAT	TADIL Analysis Tool
TCC	Test Control Center
TDAT	Test Design Analysis Tool
TDMA	Time Division Multiple Access
TDS	Tactical Data Systems
TECOM	U.S. Army Test and Evaluation Command
TIS	Test Item Stimulator
TOEL	Time-ordered Events List
USMTF	U.S. Message Text Format
VAX	Virtual Address System
VME-TIS	Versa Module Eurocard-Test Item Stimulator
VMS	Virtual Management System

Appendix C. Glossary

Command	An instruction, typed in by the user at a computer or included in a command file, which requests the software monitoring a computer or reading a command file to perform some well-defined activity.
Design	1. The specification of the working relations between the parts of a system in terms of their characteristic actions. 2. Formulating and graphically describing the nature and content of input, files, procedures, and output in order to display the necessary connection processes and procedures.
Elaboration	Currently, a scenario is made up of a scenario of timed events. Events are made up of a series of timed messages. A scenario file is created by placing each message for an event into the file according to its calculated scenario time. This process of expanding events is called elaboration.
Engineering Notebook (ENB)	A COMARCO document used to formalize technical procedures, reports, and other information in an organized fashion which can then be used to inform the government and other COMARCO personnel.
Enhancement	An improvement or addition to an existing way of accomplishing a task.
Event	In the current Pretest mode, a list of timed messages. Events do not exist in the real-time system.
Exercise Driver	A war gaming system.
Granularity	The level of detail in a system. In time measurement a high granularity might measure time in seconds, a fine granularity might measure in micro-seconds. In a scenario fine granularity defines small packets of data, high granularity would define larger packets.
Graphics	Images other than texts. Includes charts, data flow diagrams, hierarchy charts, forms, etc.
Interface	1. A connection between any two elements in a computer system. The term interface is used for the connection between items of hardware and/or software, as well as the human interface. 2. On a small computer, an electronic component, often a board, that links an external device to a computer. 3. More generally, an electronic component that links two different devices. 4. A common boundary between automatic data processing systems or parts of a single system. In communications and data systems, it may involve code, format, speed, or other changes as required.
Jump	To move forward or backward in a scenario to a specified point (identified by scenario time).

Link	1. The means of connecting one location to another for the purpose of transmitting and receiving data. 2. A channel or a line, normally restricted in use to a point-to-point line. 3. The way that a TIS-based system connects in to a network. If TIS-based system is monitoring another system, the network is accessed by tapping into another link. If a TIS-based system is stimulating another system, we would become another node on the network.
Log	A record of performance.
Log File	A file produced during test execution which contains test data (usually messages received and transmitted plus any alerts generated by the system).
Message	A communication of information from a source to one or more destinations, usually in code. A message is usually composed of three parts: 1. A heading, containing a suitable indicator of the beginning of the message together with some of the following information: source, destination, date, time, routing. 2. A body containing information to be communicated. 3. An ending containing a suitable indicator of the end of the message.
Message Format	The specific identification and placement of portions of a message, such as its heading, address, text, end of message, etc.
Message-oriented Tests	Tests designed to test protocols rather than equipment. For these test, message rates and throughput are relatively unimportant.
Micro-Test Item Stimulator (Misco-TIS)	A computer system that includes software designed to run on a MicroVAX PC.
Needline	A communications link between two modes in a system or network.
Object	In Ada, a variable or a constant. An object can denote any kind of data element, whether a scalar value, a composite value, or a value in an access type.
Object-Oriented Design	A design method that focuses on objects represented in software and on the attributes and behavior of those objects rather than on the functions that software performs.
Off-the-Shelf Software	1. Pertaining to production items that are available from current stock and need not be newly purchased or immediately manufactured. 2. Pertaining to computer software or equipment that can be used by customers with little or no adaptation, thereby saving them from the time and expense of developing their own software or equipment.

Operator	1. The person responsible for assembling and connecting hardware components and for ensuring that the computer system operates correctly. Contrast with "user." The operator and user, in some cases, may be the same person. 2. The person who actually manipulates the computer controls, places information media into the input devices, removes the output, etc.
Posttest	Occurring after the termination of test execution.
Posttest System	In TIS-based systems, the combined hardware/software system on which the data collected as log files by the Real-time System is reduced and analyzed.
Pretest	Occurring prior to the initialization of test execution.
Pretest System	1. In TIS-based systems, the combined hardware/software system on which the scenario file is designed and built for later testing by the Real-time System. 2. In the Automated Scenario Generation System, pretest will involve two phases, scenario creation and scenario execution, which would be run on separate CPUs.
Protocol	A set of conventions on the format and contents of messages to be exchanged between two or more parties. The simplest protocols define only the hardware configuration. Communications protocols define the rules for the electrical, physical, and functional characteristics of the communications link and deal with timings, data formats, error detection and correction techniques, and software structures. A communications protocol contains the control procedures required to facilitate data transfer across the link interfaces.
Real Time	Transmitting and receiving data within strict and fixed time constraints.
Real-Time System	In a TIS-based system, the combined hardware/software system on which data that are necessary to the control and/or execution of a transaction can be processed in time for the transaction to be affected by the results of the processing.
Reference	An indication of where to find specific information; e.g., by reference to a document, an author, an instruction, etc.
Response Table	A very simple protocol simulation system. When a message is received by the TIS, it refers to the response table to determine the precise response to return to the SUT.
Routine	An ordered set of instructions that may have some general or frequent use.
Scenario	In current Pretest mode, a specification of the events which are to occur for a particular test. The events are elaborated by the real-time system.
Scenario File	In the current Pretest mode, a file containing a list of timed messages for a particular test.

Scenario Generation	The process of creating and combining events to produce the required scenario file.
Script	A written set of procedures furnished to the user of a system to define step by step what is to be done to perform a specific test.
Simulate	Imitating the functioning of another system or process.
Start	To begin or resume execution of a scenario.
State	A term relating to the condition of all the units or elements of the system, i.e., the storage data, digits in registers, settings on switches, etc., including the question, "What is their state?"
Stimulate	To provide input to another system or process.
System-specific Applique (SSA)	The software that has to be written to allow TIS-based system to work for a specific system under test.
Table	A chart made up of intersecting horizontal and vertical lines to form rows and columns.
Tailoring	The making or adapting of software to fit a particular purpose, usually to meet the requirements for testing a particular SUT.
Test	To examine, particularly relative to a criterion.
Test Item Stimulator (TIS)	A computer system that includes software designed to run on a VAX mainframe.
Test Item Stimulator (TIS) Family	A set of software designed to stimulate (and on occasion to simulate) an SUT. Includes the particular platform established for running the software. The TIS thus far has evolved from the TIS to the Micro-TIS to the VME-TIS. All platforms are considered to be part of the TIS family.
Time-ordered Event List (TOEL)	A high order of granularity for scenarios.
Track	A representation of a location on an X-Y coordinate system. Used to keep track of a piece of equipment such as a plane, tank, or ship.
Traffic-oriented	Tests designed to check rate and Test throughput. In these tests, content of message is not as important.
User	The person responsible for running software and for reporting problems if the software malfunctions. Contrast with "operator." The operator and user, in some cases, may be the same person, but clear usage of the two terms is important.
User Interface	A common boundary between the computer hardware/software and the user of the application software.

Versa Module Eurocard-Test Item Stimulator (VME-TIS)	A computer system that includes software designed to run on a VME bus-based micro-computer.
Window	A software device, used in multiprocessing environments, that allows a terminal to be used to run several processes at once. The terminal screen is divided up into several windows, each of which is dedicated to a particular processor's set of processes, just as though it were an entire terminal screen.

Please use this space for your notes

Appendix D. CBS Interfaces

1. Scope

This appendix contains a description of the interfaces excerpted from a MITRE paper, WP-90200429, Automatic U.S. Message Text Format (USMTF) Text Output Message (AUTOMsgs) Requirements Specification Document, dated Octobe- 1990.

2. Excerpt

2.1 CBS Interfaces

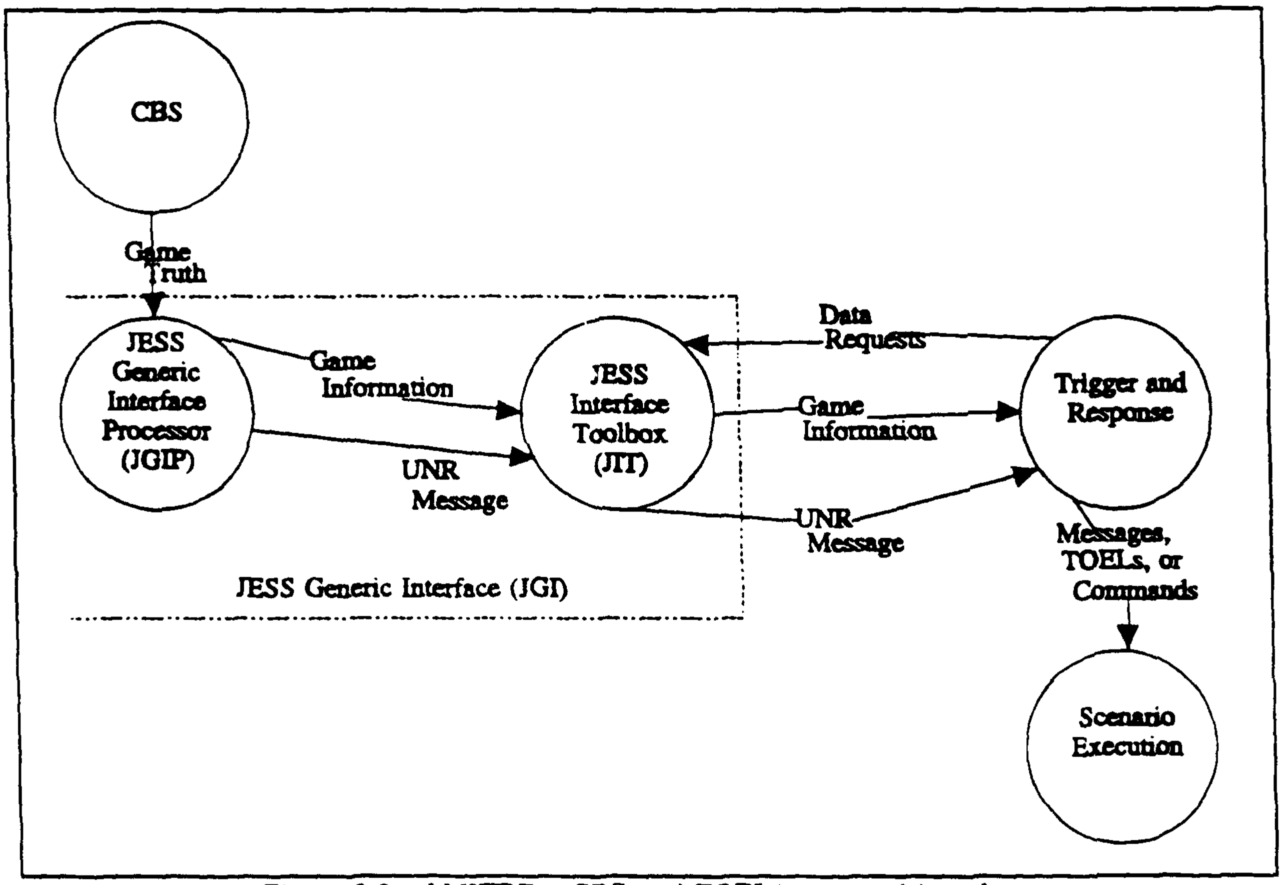

CBS
Game Truth
JESS Generic Interface Processor (JGIP)
Game Information
UNR Message
JESS Interface Toolbox (JIT)
Data Requests
Game Information
UNR Message
Trigger and Response
Messages, TOELs, or Commands
Scenario Execution
JESS Generic Interface (JGI)

Figure 2-3 (of MITRE). CBS and TOEL/command interfaces.

AUTOMsgs interfaces with CBS through the JGI described below. A secondary interface to CBS information is required by AUTOMsgs. This interface is not to CBS, but to a data file created and used by CBS. Access to the CIF is required in order to generate the USMTF ORDER message since the CIF contains specific controller orders for maneuvering game units. The following sections describe these interfaces to CBS and CBS information.

2.1.1 JESS Generic Interface

The JGI resides on an external workstation and consists of two components: JGIP and JIT. JGIP receives both game status and game information data from CBS by polling the communications interface for CBS transmissions. The other JGI component, JIT, is available as a set of C object library functions which are linked to the client application's code. JIT interfaces with JGIP to retrieve the CBS game information stored in the JGIP WS Database as well as the game time and status.

There are two potential configurations for interfacing JGI to CBS: one provides JGIP and JIT on two separate processors (MicroVax IIs) and the other places JGIP and JIT in one processor (also a MicroVax II). CBS developers generally recommend two separate processors because some client applications require a large amount of random access memory (RAM), seriously impacting JGIP communications which uses RAM for internal buffering.

As discussed earlier, CBS interfaces directly with JGIP, while JIT retrieves JGIP game data for AUTOMsgs use. Figure 2-4 illustrates how JGIP receives and stores CBS information. Three separate programs compose JGIP: Workstation Communications (WSCOMM), Update Notifications Router Receiver (UNR_RCVR), and UNR. WSCOMM provides the communications handling between CBS and JGIP, receiving game truth information from CBS and storing it in a Game Truth data store. UNR_RCVR processes game truth, updating the JGIP WS Database with game information when applicable, as well as generating a UNR message for the UNR Update Mailbox. Note that the UNR Message will not indicate the specific game object attributes, which have been updated; rather, it will indicate merely that a game object update has occurred for the game object referenced.

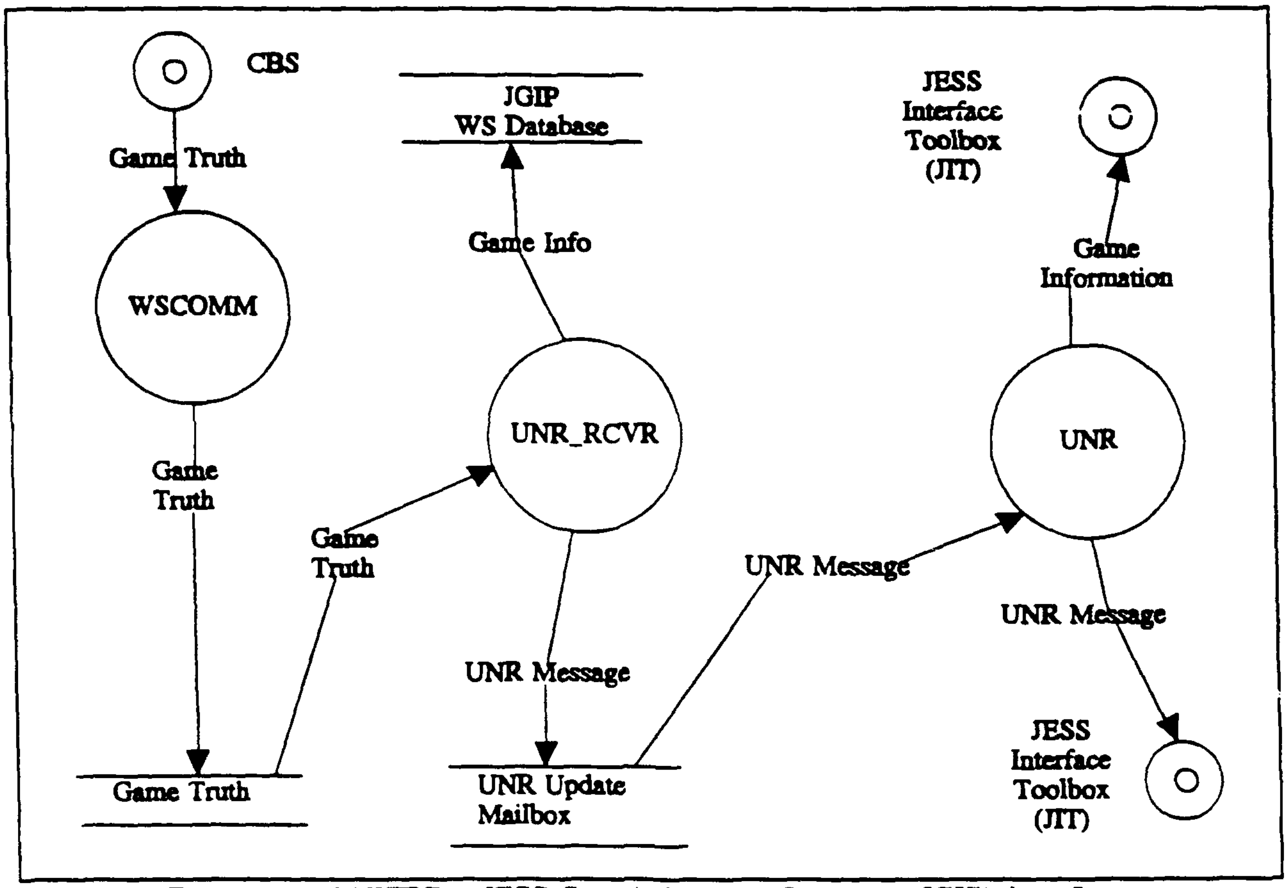


Figure 2-4 (of MITRE). JESS Generic Interface Processor (JGIP) data flows.

Finally, the UNR will send the UNR Message to the Client Mailbox for later retrieval by JIT. JIT is able to retrieve game information, game time, and game status from JGIP for an external client application's use; however, JIT does not provide any other capabilities beyond these.

Figure 2-5 illustrates the data flow between JIT and the client application, which in this case is AUTOMsgs. AUTOMsgs must initially send JIT an Attach Message to indicate its readiness for receiving game information. After that, AUTOMsgs polls JIT with a UNR Message Request until a UNR Message arrives. When JIT sends the UNR Message, AUTOMsgs may require additional information and will make a Game Information Request for the information.

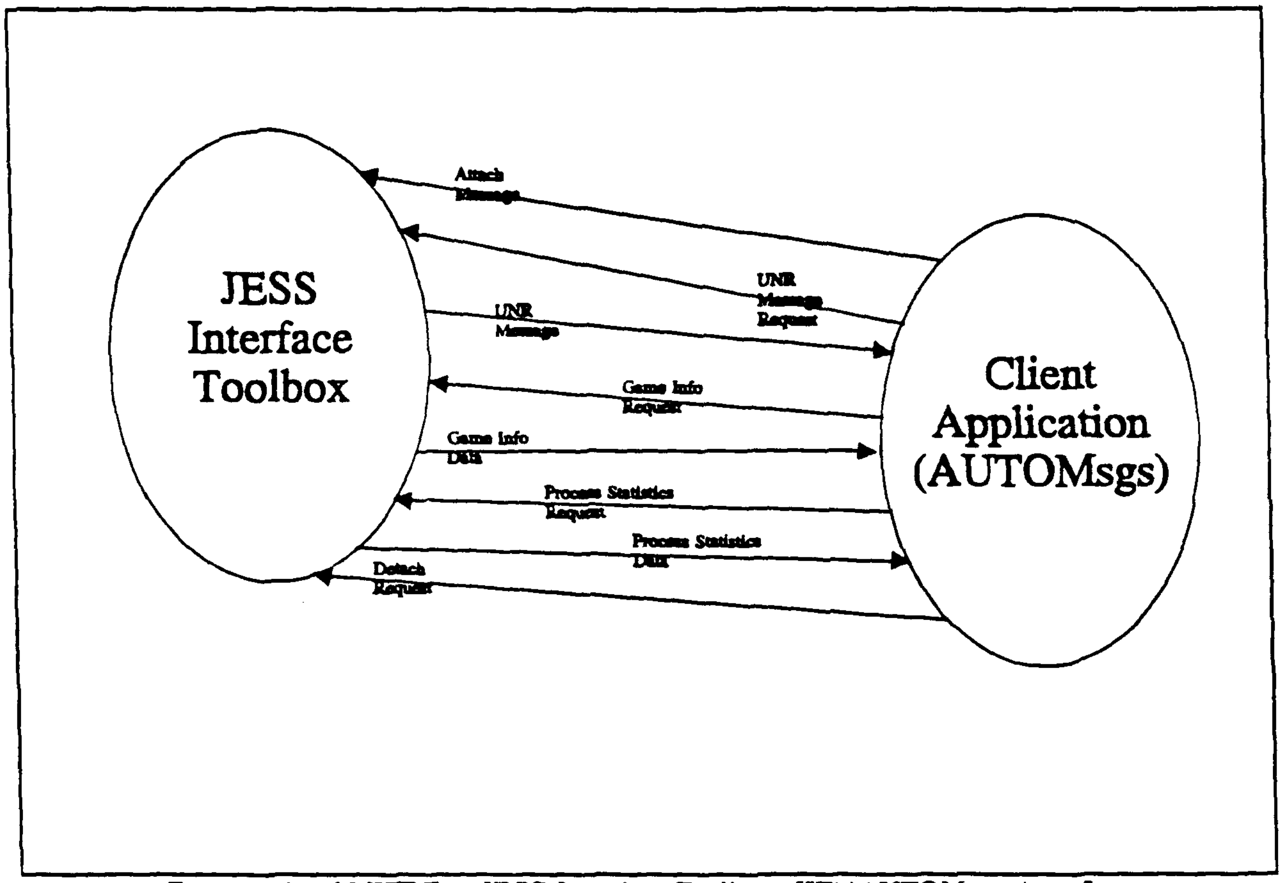


Figure 2-5 (of MITRE). JESS Interface Toolbox (JIT)/AUTOMsgs data flow.

JIT will retrieve the appropriate game object data from the JGIP WS Database, and will send it to AUTOMsgs. The Process Statistics Request and Process Statistics Data shown in Figure 2-5 provides Client Mailbox process statistics, such as JGI connection time to CBS, quantity of UNR Messages received. Finally, when AUTOMsgs wants to break its CBS connection, it sends a Detach Request to JIT.

CBS 1.2 (current version) game truth does not transmit specific scenario events to JIT; hence, AUTOMsgs never knows exactly which events are occurring. Instead, CBS provides game object update notifications which indicate that a simulation game object was changed, and AUTOMsgs must then infer what event may have occurred since the information is not provided directly. While inferencing in this manner may either cause some messages to be generated after the expected time sequence or result in differences between AUTOMsgs inferred events and CBS events. However, these kinds of differences from the game truth are typical of the fog of war for a wartime environment, thereby resulting in a more realistic test and training environment.

2.1.2 CBS Critical Input File Interface

In order to obtain the information necessary to support the USMTF ORDER message, AUTOMsgs must examine the contents of the CIF file, which is created and maintained by the CBS Game Events Executive Processor (GEEP). This file contains the orders to be executed by CBS, and may be created by CBS itself or by any of the controllers at the various controller workstations attached to CBS. The orders outline all manipulations for game objects in the simulation. AUTOMsgs will interpret some of the CBS orders for use in the synthesized USMTF ORDER message.

Appendix E. EPLRS Data Base Structures

```
Structure for database:  F:\DBASE\PLANNING\NEEDDB.DBF
Number of data records:     251
Date of last update   :  03/23/88

Field  Field Name  Type        Width    Dec    Index
    1  DEPLOY      Character       1               N
    2  NLID        Numeric         4               N
    3  SRC_MILID   Character       8               N
    4  SRC_TYPE    Character       5               N
    5  SRC_LCN     Character       2               N
    6  DST_MILID   Character       8               N
    7  DST_TYPE    Character       5               N
    8  DST_LCN     Character       2               N
    9  PRI         Numeric         1               N
   10  RATE        Numeric         1               N
   11  ACK         Logical         1               N
   12  CLASS       Character       1               N
   13  MSG_HR      Numeric         3               N
   14  BITS_MSG    Numeric         5               N
   15  SOS         Numeric         2               N
   16  SRTKEY      Character      21               N
   17  VALID       Logical         1               N
** TOTAL **                       72
```

```
Structure for database:  F:\DBASE\PLANNING\UNITDB.DBF
Number of data records:     515
Date of last update   :  03/22/88

Field  Field Name  Type        Width    Dec    Index
    1  DEPLOY      Character       1               N
    2  UNIT_NO     Character       4               N
    3  MILID       Character       8               N
    4  TYPE        Character       3               N
    5  EAE         Character       3               N
    6  MSG_SETS    Character      12               N
    7  GROUPS      Character       3               N
    8  HOME_UNIT   Character       4               N
    9  CLASS       Character       1               N
   10  GARR_GRP    Character      35               N
   11  DESCRIPT    Character      25               N
   12  TASK_GRP    Character      35               N
   13  SITE        Numeric         3               N
   14  HOST        Character      12               N
   15  SIM_TYPE    Character       5               N
   16  HOST_SLOT   Character       1               N
   17  HOST_CHAN   Character       1               N
   18  INTERFACE   Character       9               N
   19  URO_SLOT    Character       1               N
   20  URO_CHAN    Character      1              N
   21  BPS         Numeric        6              N
   22  HOST_CHAR   Character      1              n
** TOTAL **                     175

Structure for database:  F:\DBASE\PLANNING\SITEDB.DBF
Number of data records:    1000
Date of last update   :  03/22/88

Field  Field Name  Type         Width    Dec    Index
    1  SITE        Numeric          3               N
    2  EAST        Character        6               N
    3  NORTH       Character        7               N
    4  ELEV        Character        4               N
** TOTAL **                        21
```

Appendix F. Existing Scenario Generation Software for MSE

1. Scope

This appendix deals with existing MSE scenario generation software.

2. Existing Scenario Generation Software

2.1 Basic Pretest Software. The existing Pretest software for MSE runs on the TIS. There are difficulties in running on that platform because of the amount of time it takes to generate scenarios. Pretest software is described in detail in ENB 45-6-1.1 A, TIS Pretest Operator's Manual for MSE, dated 1 November 1989.

2.1.1 Functions. The following subparagraphs describe the seven basic functions performed by the Pretest software.

a. Test Creation. An ASCII file is sent to the VAX and is appended to the event format library (EFL) to create events. Then test events are configured and specified, and scenarios are automatically created. The initialization tables and load commands are also automatically created.

b. Test Specification. This function allows the user to enter the configurations of the scenarios. Each configuration consists of the system-specific applique processor (SSAP), the SSA, the real-time variables, the system names, and the scenario name. This allows the real-time user to see how the test should be set up.

c. Scenario Creation. This function allows the user to build definitions of message streams. These streams are made up of events from the event file library. The user arranges these events in time-sequenced order. The user can then set the parameters needed at the start of the real-time test. Parameters include scenario notes, table initialization of SSA subscribers by auto affiliation, and SSA load tables to initialize the SSA at the start of the real-time test. The scenario creation function must be run on the events before real-time file generation can be run.

d. Real-time File Generation. This function allows the user to generate a scenario file for a scenario produced by the scenario creation function. The scenario file will contain the bit stream that will be presented to the SSA. The user will be prompted to enter the Virtual Memory System (VMS) file name for the scenario file.

e. Event File Generation. This functions allows the user to generate individual events to be merged in the real-time scenario file. Events consist of bit stream messages specified in the EFL.

f. Scenario Preview. This function allows the user to look at the file as it will be sent to the SSAP. During the scenario preview, the user can choose the type of output, the format of the output, and where the output is to appear. The type of output can be messages (from a scenario) or track initialization data. The format can be dump or statistics.

g. Event Format Library. The event format library consists of many different types of fields. Combinations of those fields are called messages. Combinations of those messages are called events.

(1) Each field in the library consists of a value and the following parameters:

- Field name.
- Data base type.
- Conversion type.
- Conversion bytes.
- Conversion bits.
- Scale factor.
- Field description.

NOTE: For some types of fields, the user-entered values are coded into bits.

(2) Each message in the library consists of one or more fields and the following parameters:

- System name.
- Message name.
- Message type.
- Message code.
- Message description.
- Form name for the message.
- Posttest cross-reference.
- Field name.
- Byte location.
- Bit location.

NOTE: The first six parameters describe the message and the next six parameters describe how the fields of a message are put together when the message is transmitted.

2.1.2 Screen Information. The following information was obtained from Appendix I, Event Format Library Generation in ENB 45-6-1.1 A, TIS Pretest Operator's Manual for MSE, dated 1 November 1989.

a. Message Event Generation Screen. This screen generates messages for the message event specification. The messages generated will be of the same type and will have the same message ID. The input fields are as follows:

- Event length (1-60).
- Message loading function (linear-L or exponential-E).
- Message rate start (1-999999 messages per hour).
- Message inter-transmission interval (fixed-F or random-R).
- Msg ID (8 characters).
- Seg (segment number 0-999).
- Destination (channel number 0-99).
- Msg Error (must be blank).
- Staleness (0-3600000).
- Spec Proc Flag (must be blank).

b. Message Data Generation Screen. This screen allows the user to modify any control field or data field for all messages in the event with a specified ID. The data generation form displays a list of all the message IDs currently in the event and the count for each. The parameters are as follows:

- RTIndx1.
- RTIndx2.

- Segment.
- Destination.
- Msg Error.
- Staleness.
- Conversion Flag.
- Field Seq#.

c. Message Event Specification Screen. This screen contains an entry for each message event specification currently found in the data base. The form is displayed in either the update or read mode depending on whether changes are allowed, and that depends on whether or not the event is referenced by a scenario.

(1) Although this screen deals with field definitions, message definitions, and event definitions, MSE does not use the field and message definition portions.

(2) The event definition parameters are as follows:

- System Name.
- Event Name.
- Event Description.
- Event Type (which includes the following subparameters):
  - Destination.
  - Segment Number.
  - Staleness.
  - Message Errors.
  - Special Processing Code.

NOTE: The event type subparameters describe how a message is to be transmitted in an event.

2.1.3 Data Entry. After a message is entered into an event, the user can enter data directly into the message.

2.2 TDAT Software. The existing TDAT software for MSE runs on a PC rather than on a TIS. It was developed in response to a need for a faster way to generate scenarios than using the Pretest software on the TIS platform. TDAT operations are described in detail in ENB 45-6-4.2 ORIG. Test Design and Analysis Tool User's Manual, dated 5 March 1990.

NOTE: The existing TDAT software was to be used (at least to some degree) during the MSE OY4 test. MSE OY4 scenario generation is described in more detail in paragraph 2.3 below.

2.2.1 Programs. TDAT is a collection of programs (including TDAT, CRDBCOPY, MAKTEST, plus 11 other programs) that perform several functions. Alternative methods exist for each function, and sometimes only some of the programs are used. TDAT is written in C, CRDBCOPY and MAKTEST are written in PASCAL, and the remainder are written in dBASE III+. A separate file is used for each hour. Some TDAT programs only work on one hour's file at a time, while others work on files for all hours.

2.2.2 Problems

a. The TDAT process involves many steps, many data formats, many software programs, and is run on two different platforms. There appears to be as much of an information representation/data base problem as an information processing problem. Possibly a better information representation would allow data validation/checking as information is input rather than later on.

b. The output of TDAT is not just one file, it is a collection of the following files:

- .TIS file.
- .SCN file.
- Event library.
- Scenario index.
- Newdat.
- Scendb.
- Specify table.
- Load command table.
- Init table.

2.2.3 Procedures for Generating a Scenario. The following steps are used to generate a scenario.

a. Create a TIS file in the format defined in ENB 45-6-4.2 ORIG, Test Design and Analysis Tool User's Manual. That format is provided in Table 1

Table 1. Messages for MSE OY4 Test

CODE a	MSG #	SHORT TITLE	MAX SIZE (CHARS)	AVG SIZE (CHARS)
3	C110	INTREP	2257	500
4	C111	TACREP	3458	800
5	C400	SITREP	5042	1200
6	F014	REQ INFO (RI)	1291	300
7	F015	RESP TO RI (RRI)	863	200
8	G131	INTSUM	6354	1600
9	G880	PERSTAT	695	200

b. Create field definitions.

c. Create and specify event definitions.

d. Create scenarios.

e. Add events to the scenarios.

f. Initialize tables.

g. Add notes to scenario.

h. Create and specify test index.

i. Generate a real-time file for a scenario.

j. Preview a scenario.

k. Modify a generated scenario.

Appendix G. General Description of VME-TIS System

1. Scope

This appendix provides a general description of the VME-TIS system as it is currently being developed.

2. Description

The VME-TIS is the latest member of the TIS family of digital data link drivers. Those drivers are used to stimulate and test the digital data links of C^3I systems. TISs substitute for the systems with which the C^3I systems communicate. The TISs provide a total data-link test-environment as illustrated in Figure G-1.

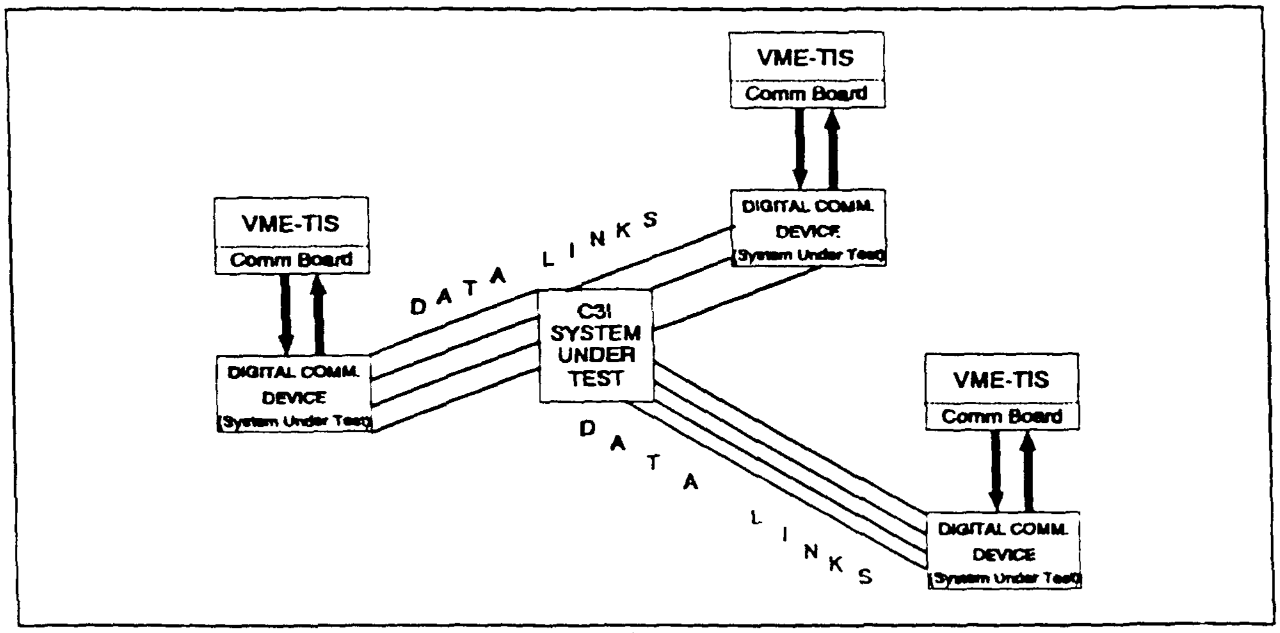


Figure G-1. Data link test environment.

The VME-TIS is based on modules plugged into the VME instrumentation bus. Its modules include the computer processor, memory, disk drive, time code interface card, and front-end communication boards as shown in Figure G-2. The special front-end boards physically and electrically connect the VME-TIS to the C^3I SUT. These front-end boards handle the protocol processing for the TISs by taking the bits apart on messages coming in and putting the bits together for messages going out.

For the TIS family, software consists of a generic part and an SSA. The SSA software is tailored for each application. The TIS family's software has been used to support testing of JTIDS and MSE.

SSA software used in those applications could be rehosted to the VME-TIS. In addition, SSA software for ADDS, EPLRS, and JDRA is being developed. The VME-TIS could have many applications in ATCCS and associated BFA testing.

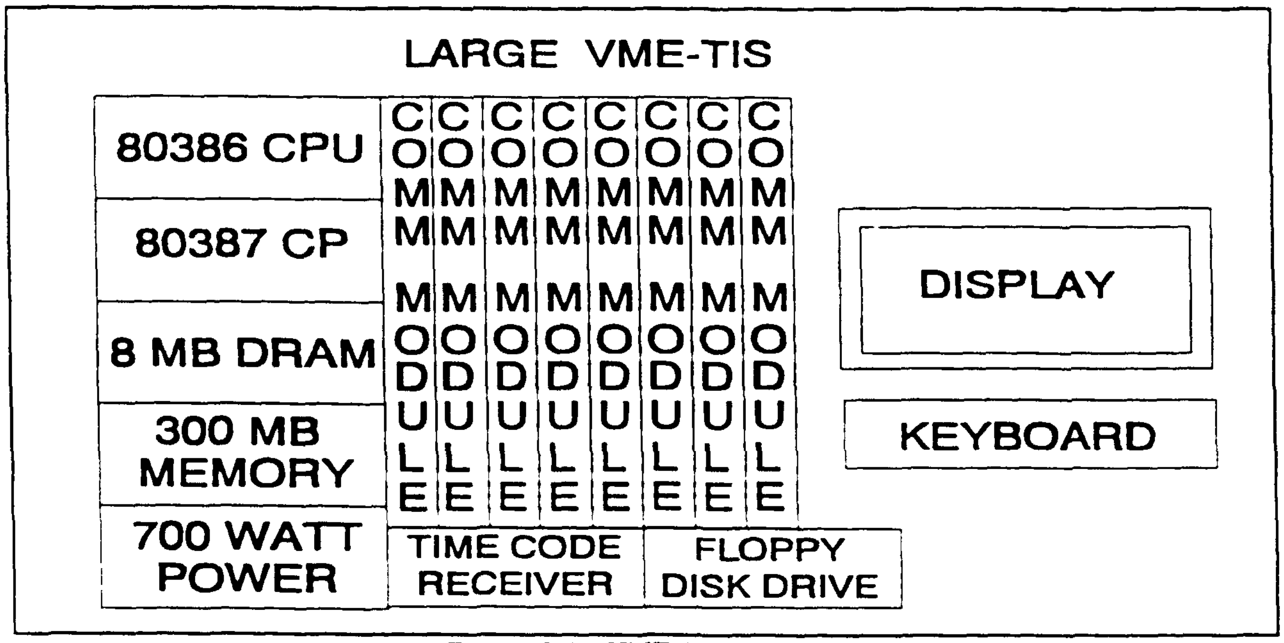


Figure G-2. VME-TIS Modules.

Appendix H. VME-TIS Communications Module

1. Scope

This appendix provides a description of the communications module currently being developed for use with the VME-TIS system and is illustrated in Figure H-1.

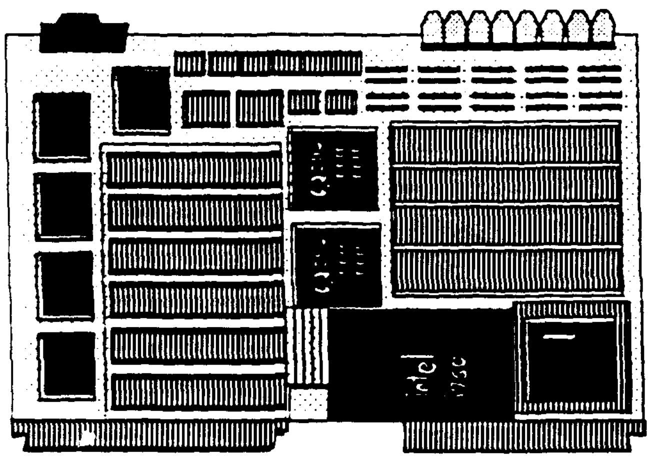


Figure H-1. Communications module.

2. Description

The communications module connects the VME-TIS to a digital communications device as shown in Figure H-2. This connection is needed so the VME-TIS can send message traffic to the communications device. This connection also allows the VME-TIS to receive messages and to monitor an existing link between communications devices.

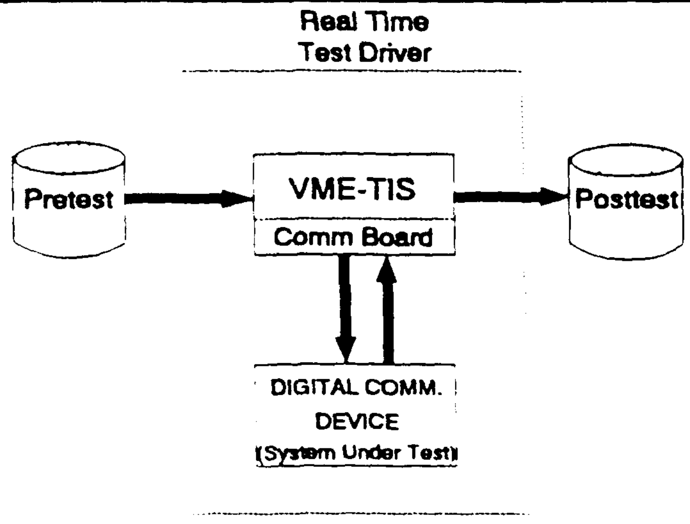


Figure H-2. Communications board interfaces.

The communications module records information, which it sends to the VME-TIS processor for storage. This information can reveal what happened in the test environment and it includes message content, time of transmittal, and time of arrival. Through analysis, this information can yield system-specific information such as speed of service and throughput for a given link, or it can baseline the performance for future tests.

The communications board contains a daughter board that does the electrical interface as shown in Figure H-3. For received messages, this means converting the electrical signal into bits (ones and zeros). The board will support various standards, including the following: MIL-STD-188-114A, RS422/423, RS232C, and MIL-STD-1553B.

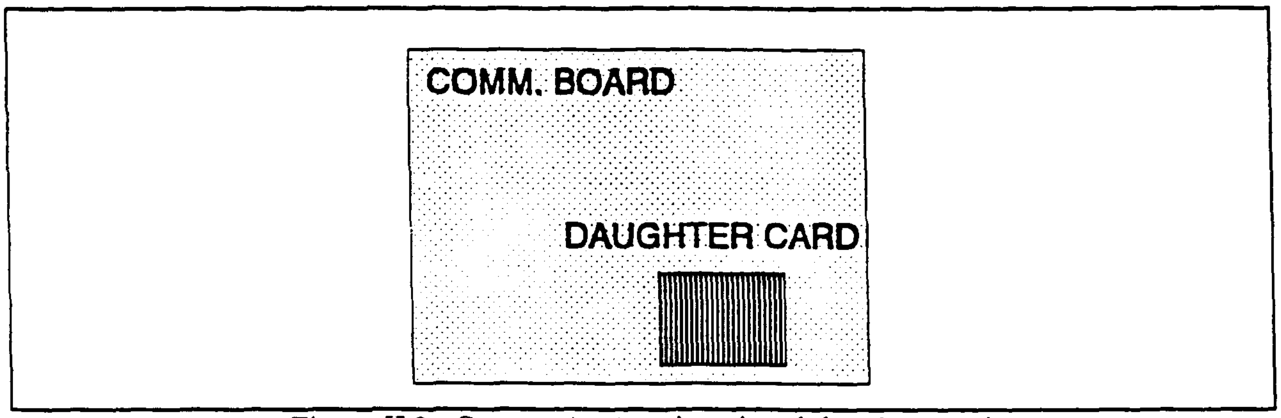


Figure H-3. Communications board and daughter card.

Next, the board does error detection and correction, strips out the bits meant for network control/addressing and protocol, and records on the board the message information with time information from the time code receiver. Periodically, the stored information is sent to the VME-TIS processor for permanent storage.

The board's outstanding features are high-speed multi-link protocol processing and accurate time stamping. Each board will provide a total processing speed of 6.6 millions of instructions per second (MIPS) over 4 channels. The small VME-TIS driver will contain 2 boards and will support up to 8 channels. The large VME-TIS driver will contain 8 boards and so will support up to 32 channels. Each channel can be up to 512 Kilobits (Kbits) full duplex.

Appendix I. Extract from IDD for ADDS Implementation of VME-TIS

1. Scope

This appendix contains an extract from the draft version of IDD-7A-10-00-00-OR, Interface Design Document for the ADDS Implementation of the Versa Module Eurocard-Test Item Stimulator System, dated 10 May 1991.

2. Extract

3.2 Scenario Records. The interface between the Pretest CSCI and the Real-Time CSCI is a group of scenario records. Their purpose is to transfer the scenario data to the Real-Time CSCI for use in stimulating the SUT. The data is transferred on one or more diskettes or on an 8-millimeter tape to the Real-Time CSCI in the form of a scenario file. The Real-Time CSCI uses this file as a sequential file in Microsoft-Disk Operating System (MS-DOS) format. Figure 3.2-1 illustrates the layout of a scenario file. Figure 3.2-2 shows the general layout shared by all scenario records and the basic structure of each record type. Each record in the scenario file contains a data length in bytes. The data length is used to calculate the location of where to begin reading the next record from the scenario. Since there are scenario records for only a few System-Specific Applique Processor (SSAP) commands (i.e., Auto Load, Stop, Force Save), and there are log records for all SSAP commands, basic structures of SSAP commands will be included in the paragraphs describing the log records.

3.2.1 Scenario Record Data Elements. Table 3.2.1-I details data to be used in the records of the scenario file. The table columns are:

a. Data Element - Name (or mnemonic) of the shared data element.

b. Description - Brief description of the shared data element.

c. Data Type - Type of shared data being transmitted expressed in terms of the programming language, e.g., integer, real, string logical, complex, etc.

d. Data Representation - The manner in which the shared data element is represented, e.g., hex, octal, American Standard Code for Information Interchange (ASCII), single precision, double precision, floating point, fixed point.

e. Data Source - The file, packet, record, etc., from which the data is obtained.

f. Location Where Used - Name of the computer software component (CSC) where the data element is used.

g. Limit/Range - The limits or ranges of acceptable local data values, e.g., T or F, Y or N, 0 or 1, On or Off, CH001-CH007.

h. Units of Measure - The kind of measurement indicated by the data element (if applicable), e.g., distance (in miles), speed (in miles per hour or knots), coordinates (in degrees and minutes of latitude and longitude), or time (in hours, minutes, and seconds).

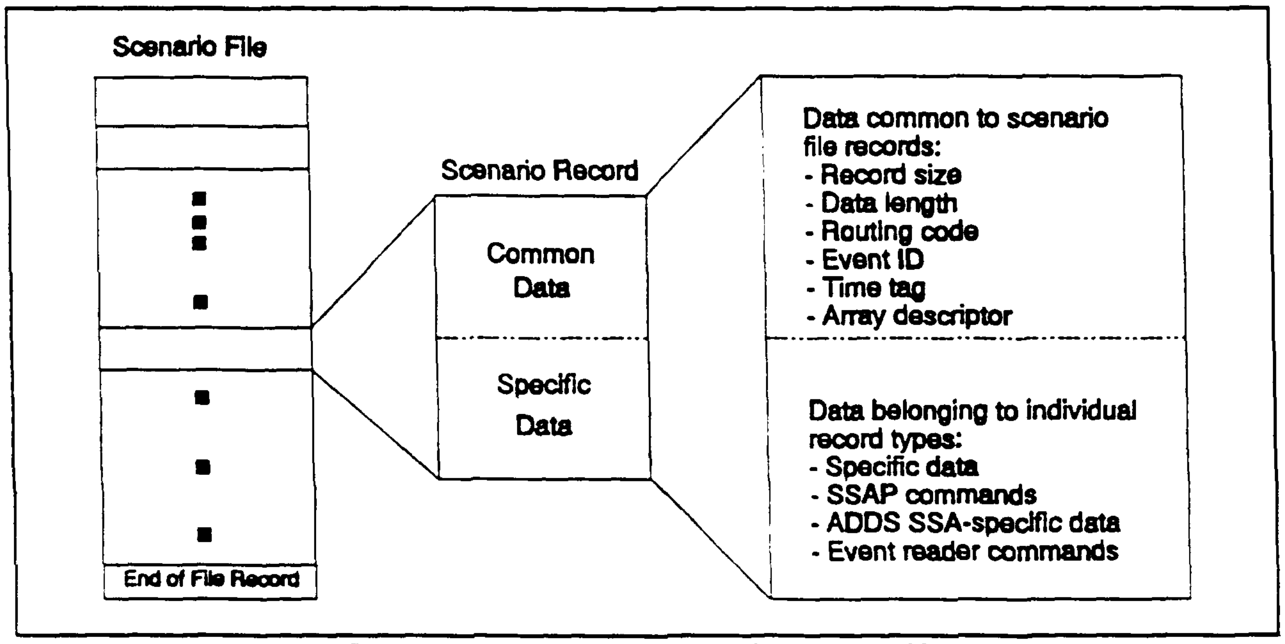


Figure 3.2-1 (of IDD). Layout of a scenario file.

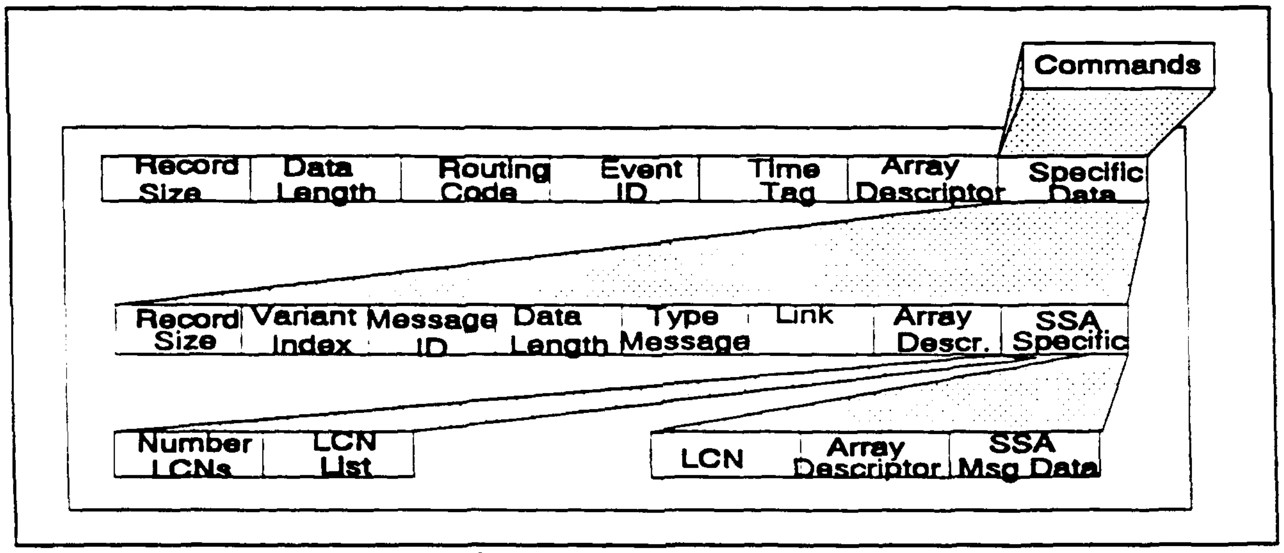


Figure 3.2-2. Basic structure of scenario records.

3.2.2 Scenario Record Data Structure Descriptions. The data structures within the scenario file are described in Table 3.2.2-I. Figure 3.2-2 shows data items contained in each data structure. Table 3.2.2-II lists the data items in the scenario file in alphabetical order, and shows each scenario record that contains the data item. Data that is contained in all types of scenario records is designated 'common data.'

3.2.3 Scenario Record Interface Priority. Since the format for the scenario record interface is a transferred file, which is loaded from the pre-vtest facility for the Real-Time VME-TIS before the test is begun, interface priority is not applicable to the scenario record interface.

3.2.4 Scenario Record Communications Protocol. The scenario records shall be communicated from the Pretest CSCI to the Real-Time CSCI of the VME-TIS as follows:

a. The scenario records shall be written as a file. The file shall be converted to the MS-DOS operating system format, i.e., a format that is immediately readable in the Real-Time mode.

b. After the conversion, the scenario file may be written onto one or more 3.5-inch floppy diskettes or an 8-millimeter tape containing the scenario file. The diskettes or tape shall be hand-carried to the Real-Time HWCI.

Table 3.2.1-I (of IDD). Scenario Record Internal Data Elements

Data Element	Description	Data Type	Data Representation	Data Source	Location Where Used	Limit / Range	Units Of Measure
Type of Message	place where message generated	integer	whole numbers	scenario file	SSA	scenario, operator RT, response, receive	NA
SSA Data Length	lenght of SSA data	integer	whole numbers	scenario file	SSA	physical limits	bytes
Link	number of physical links for a message	integer	unsigned value	scenario file	SSA	1-32	NA
SSA Array Descriptor	describes array with message data	integer	whole numbers	scenario file	SSA	NA	NA
Number of LCNs	number of logical channels ADDS unit	integer	whole numbers	scenario file	SSA	1-45	NA
LCN List	list of logical chnnel numbers used ADDS	array	byte value	scenario file	SSA	45 values between 0-255	NA
LCN	number of logical channels where ADDS message directed	byte	unsigned value	scenario file	SSA	0-255	NA
T/R Array Descriptor	describes array with ADDS data	integer	whole numbers	scenario file	SSA	NA	NA
Message Data SSA	Data to be used by ADDS unit under test	string	variable	scenario file	SSAP	NA	NA

Table 3.2.2-I (of IDD). Descriptions of Scenario Records

RECORD	DESCRIPTION
SSAP Command	Contains commands to Real-Time tasks for processing during the test
SSA-Specific Data	Contains messages to be sent to the ADDS unit being tested
Event Reader Command	Contains commands to be processed by the Event Reader

Table 3.2.2-II (of DTT). Data Items in Scenario Records

DATA ITEM	SCENARIO RECORD STRUCTURE
ADDS SSA Message Data	SSA-Specific Data
Array Descriptor	Common Data, SSA-Specific Data
Data Length	Common Data, SSA-Specific Data
Event ID	Common Data
LCN	SSA-Specific Data
LCN List	SSA-Specific Data
Link	SSA-Specific Data
Message ID	SSA-Specific Data
Number of LCNs	SSA-Specific Data
Record Size	Common Data, SSA-Specific Data
Routing Code	Common Data
Time Tag	Common Data
Type of Message	SSA-Specific Data
Variant Index	SSA-Specific Data

Appendix J. Extract from Instrumentation Validation Test Plan for TCC

1. Scope

This appendix contains an extract from IVT-02-OR, Instrumentation Validation Test Plan for the Mobile Subscriber Equipment, dated 31 July 1991. The information in the extract pertains to the functioning of the TCC currently under development.

2. Extract

1.4.1 Test Control Center.

a. The function of the TCC is to provide real-time test control capabilities, allowing the Test Director to continuously view the status of the test and to rapidly identify problems as they occur. To affect this kind of control remotely in situations where assets may be mobile, a flexible radio-based data communications capability is required. Additionally, the interface provided to the Test Director must be graphically oriented and relieve the Test Director of having detailed knowledge of the computer or communications systems employed. Both of these capabilities have been provided by the TCC.

b. The TCC consists of two Sun IPC work stations and a variable number of packet radio networks. One of the work stations provides control of the test instrumentation through the packet radio networks and receives performance data in return (Figure 1.4.1-1). <Figure withdrawn> This work station graphically displays the connectivity of the MSE network test configuration both logically and physically. The second work station contains all information which has been received from each instrumented item. Each analyst can utilize this work station to review selected data of interest.

1.4.1.1 TCC Displays.

a. The test control work station will provide the Test Director with a number of displays, all of which are oriented to display pertinent information on the test status at each stage of the test. Initially, the TCC will display a checklist which will allow the director to check-off those stations which have reported successful completion of setup. Once setup has been verified for all test items, the establishment of the packet radio network to each item will be shown graphically as connections to each instrument are made. When the control network is operational, the Test Director will be able to display the logical or physical layout of the system under test (SUT). The order of these displays and any display may be selected at any time.

b. The MSE network connectivity diagram is a logical layout of the SUT, based on validated test configuration, and displays the network connectivity as a color-coded status for each trunk based on information received from the NC LDEDs. The MSE network geographic diagram displays the location of MSE elements as an overlay onto a map of the test range. The MSE network performance display is a textual, tabular formatted display of pertinent performance data, segregated by type of stimulator. Additionally, there are a number of engineering displays which allow on-line troubleshooting of the control network. The second work station is networked to the first through a thin wire Local Area Network (LAN) connection and provides quick-look analysis capabilities.

Please use this space for your notes

Appendix K. Study of MSE System for OY4 Testing

1. Scope

This appendix contains the results of a study made of the current MSE system to determine what changes needed to be made to scenario generation prior to the OY4 testing of MSE.

2. Objective

Software for the testing was to be completed in late June. The data message scenarios (both X.25 and LAN) were to be created using that software. Our objective was to develop an understanding of the OY4 methodology for scenario generation.

3. Earlier Investigation

An investigation of scenario generation had been conducted for MSE by COMARCO, Inc. The results are included in the following subparagraphs.

3.1 Background. In the past, USMTF messages have been individually prepared entirely by manual means. When scenarios contain hundreds of messages, the task was manpower intensive. Although there may have been other types of auto-message generators, to our knowledge, none were data base driven.

3.2 Purpose of Investigation. The purpose of the investigation was to determine the feasibility of designing a data base-driven automated generation software for use on the Micro-TIS. The data base would then be used to auto-fill all (or as many as possible) of the fields in the USMTF message.

3.3 Data Bases. The data bases used would have to be extensive if they are to contain all the possible data entries for each field of a message, but it might be possible.

3.4 Use.

a. If the software could be linked with CBS, as the CBS battle progresses and events are sent to the automatic message generation software, that software would generate the appropriate message(s) for that event with little human intervention.

b. In order to determine the scope of the task, the message formats studies were those used by the MSE. However, this data base-driven automatic scenario generation software could be used in various future system tests, particularly in the ATCCS arena.

4. Test Planning

The focus of the current test planning for MSE seems to be on the multifaceted use of instrumentation, with special emphasis on its use and interface with combat models. MSE is seen as a developer's tool with future uses as a training device and a readiness evaluation tool.

The types of commands to be used are changing. The current commands are resume, hold, pause, and stop. For MSE OY4, commands will be multi-scenario, variable message rate, and off-hook percent. Future control will deal with posture codes, attrition, location, and resupply.

During future efforts, emphasis will be placed on finding ways for the scenario generation process to support more realistic field testing. Interaction between test control and scenario generation will be required. Generating a scenario and modifying it during real time can be used to control a test or to respond to a new test environment (a dependent or independent variable).

5. CBS INTERFACE

Interfacing with CBS may be a major consideration. Interfacing may occur at any of the following levels:

a. Run CBS and use CBS statistics to generate random traffic.

b. Run CBS and keep a detailed script of when messages are sent. Use that script as the scenario.

c. Run CBS concurrent with the field test. Use some coarse values, such as posture, as input to a message generator.

d. Run CBS concurrent with a field test. When CBS sends a message, use a message generator to send the equivalent message in the test. The message generator will then get feedback on whether the message got to its destination and will provide that feedback to the CBS, which will update its war game appropriately.

6. INSTRUMENTED CALL SCENARIOS

Three kinds of instrumented call scenarios were required for the test -- Node Center, Small Extension Node (SEN), and Mobile Subscriber Radio Telephone Terminal (MSRT).

6.1 Manual Creation. Each of the three kinds of instrumented call scenarios were being produced by an MSE expert using pencil and paper.

a. The resulting scenarios would generate call information for a 30-minute block of time. Call information would include source, destination, call start time, and length of call.

b. The scripter repeated the process to describe three sets of calls (15 percent off-hook, 20 percent off-hook, and 25 percent off-hook).

c. The process was made more complex by system-specific characteristics. For example, each call was either local, semi-local, or non-local.

6.2 Automated Creation. A program was being developed to automate two parts of the process. dBXL, which uses a dBASE-compatible data base and has extra user functions, and Quicksilver, which compiles programs to increase speed, were being used for the development.

a. The parts to be automated were assignment of instrument numbers and the changing of the length of non-local calls, which changes the off-hook percentage and is a key test parameter.

b. It was anticipated that the instrument numbers would not be provided until just before the start of test and that the off-hook requirements could change just before the start of test. Therefore, automation of those processes was deemed necessary.

6.3 Data Message Scenarios. Code was being written to allow the creation of the data message scenarios for MSE OY4. The code is in FORTRAN and will run on the Micro-TIS. The scenario generation and scenario changing capability will be simpler and optimized compared to the capability of TDAT used in the past.

a. The new scenario generation process will generate the following information for each message:

- Type.
- Origin.
- Destination.
- Start time.
- Length.

b. During test execution the SSA takes that information, generates text for messages, and sends messages.

NOTE: This differs from existing scenario generation systems because they create an entire text for each message before test execution. In the MSE OY4 test, message content does not need to be unique because it is more of a technical test than a simulation test.

c. Planned for this effort is the capability to change process calls for the message rate and changing the mix of message types based on posture and situation. An example of posture would be a river crossing. An example of situation would be 50 percent attrition of all units.

d. For this effort a simple TOEL, consisting of seven types or messages, would be established. Table K-1 below summarizes the message types.

Table K-1. Messages for MSE OY4 Test

CODE	MSG #	SHORT TITLE	MAX SIZE (CHARS)	AVG SIZE (CHARS)
3	C110	INTREP	2257	500
4	C111	TACREP	3458	800
5	C400	SITREP	5042	1200
6	F014	REQ INFO (RI)	1291	300
7	F015	RESP TO RI (RRI)	863	200
8	G131	INTSUM	6354	1600
9	G880	PERSTAT	695	200

Please use this space for your notes

Appendix L. Information on TADSS

1. Scope

This appendix contains information on TADSS, a system currently under development in association with the TCC.

2. Phases

In Phase I, TADSS will be a static tool used by the experts. Initial operating capability (IOC) is scheduled for January 1992. In Phase II, some dynamics will be added and Artifical Intelligence (AI)/expert systems will be added. In Phase II, TADSS will interact with a combat model.

3. Test Situation

The TADSS will be used in a test situation in which test sites interact with Combat Electronic Warfare Intelligence (CEWI) devices based on scripts.

The test sites monitor user actions and based on those actions provide the user with information in various forms. The TCC will provide basic commands to the test sites, such as start, stop, and play track number 44.

It has not been decided what information, if any, the TCC or the test site will be recorded for analysis.

Further information on the test situation is available in the Intelligence and Electronic Warfare Training and Evaluation Complex (IEWTEC) Design Report (For Official Use Only), dated 27 August 1990.

4. Functions

The complete functions will include the following:

- Deployment building.
- Battlefield display.
- Graphical user interface.
- Interfaces to combat models.
- Model dynamics.

The functions will be handled by one of the following three portions of the software:

- User interface management system for deployment/scenario building and battlefield/graphic display.

- Core engine that communicates with various models.

- Data storage/access system for both generic data and scenario data.

5. Case Tool

The TADSS development will use a CASE tool called Software Through Pictures to do a structured analysis and create object oriented design. That tool will generate Ada package specifications and stub specifications, which can be used by Ada programmers to develop the code.

6. Script Generation Process

The preliminary design of the script generation process was as follows:

a. Create a layout using TADSS.

b. Create a master script of what combat action happens manually.

c. For each master script event, add appropriate audio, video, and human intelligence from a canned data base. (It will be necessary to create each data base item the first time it is needed.)

d. Based on the layout, add RF emitters manually.

e. For each RF emitter, add its characteristics using TADSS.

f. For each test site, create a data base with those items relevant to that test site.

g. For each test site, create a script with changes to that data base for each one-second window. (Both the time of each change and the new value of each change are included in the script.)

h. For each test site, create tracks that can be played at TCC control.

Appendix M. References

1. DCN 1, Interface Design Document for the ADDS Implementation of the Versa Module Eurocard-Test Item Stimulator System, DRAFT, 16 August 1991.

2. DOD-STD-2167A, Military Standard, Defense System Software Development, 28 February 1988.

3. ENB 45-6-1.1 A, TIS Pretest Operator's Manual for MSE, 1 November 1989.

4. ENB 45-6-4.2 ORIG, Test Design and Analysis Tool User's Manual, 5 March 1990.

5. IDD-7A-10-00-00-OR, Interface Design Document for the ADDS Implementation of the Versa Module Eurocard-Test Item Stimulator System, DRAFT, 10 May 1991.

6. IVT-02-OR, Instrumentation Validation Test Plan for the Mobile Subscriber Equipment, 31 July 1991.

7. MIL-STD-188-144A, Electrical Characteristics of Digital Interface Circuits, 24 March 1976.

8. MIL-STD-1553B, Aircraft Internal Time Revision Command/Response Multiplex Data Base, with Notice 2, 22 April 1988.

9. RS 422/423, Electrical Characteristics of Balanced/Unbalanced Voltage Digital Interface Circuits, December 1978.

10. TECOM 70-17, U.S. Army Test and Evaluation Command (TECOM) Regulation

11. RS 232C, Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange, August 1969.

12. Unnumbered, Intelligence and Electronic Warfare Training and Evaluation Complex (IEWTEC) Design Report (FOR OFFICIAL USE ONLY), 27 August 1990.

13. WP-90W00429, Automatic U.S. Message Text Format (USMTF) Text Output Messages) (AUTOMsgs) Requirements Specification Documents, MITRE, October 1990.

Please use this space for your notes

Appendix N. Distribution

Agency	No. of Copies
Commander U.S. Army Test and Evaluation Command ATTN: AMSTE-TC-S 2 ATTN: AMSTE-TC-D Aberdeen Proving Ground, MD 21005-5055	5
Commander Defense Technical Information Center ATTN: FDAC Cameron Station Alexandria, VA 22304-6145	2
Commander U.S. Army Electronic Proving Ground	
ATTN: STEEP-TD	1
ATTN: STEEP-TS	1
ATTN: STEEP-MT	1
ATTN: STEEP-MO	2
Fort Huachuca, AZ 85613-7100	

Please use this space for your notes